THE UNIVERSITY
of EDINBURGH

A virtual reality computer program for biomolecule structure determination

Hamish Todd

Submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
2019

*"**Instruments are at hand***
which, if properly developed,
will give all people
access to,
and command over,
***the inherited knowledge of the ages**"*
~Introduction to "As I May Think", 1945

*"**Everything has benefits, everything has costs; you have to pay attention to the costs.***
The benefits are obvious because they are why the thing got proposed.
***The costs tend to be subtle, or are swept under the rug by the people who want to sell you on the benefits**"*
~Jonathan Blow

Acknowledgements

I would like to thank Andrew Goryachev for his optimism about the hardware I both became interested in, and his willingness to allow me to experiment. I thank Jeyaprakash Arundalam for helping the project to avoid the unfocussed fate of many blue-skies VR projects, and for helping introduce me to the main player in my project, Paul Emsley. I learned from Paul, over the course of many conversations, in the main taking place over fortunately long car journeys, enormous amounts about structural biology, and about the needs of scientific software.

For stimulating and informative conversation I am indebted to many people. On the quantitative/software side, Ivan Erofeev, Ivan Marychev, Pontus Granström, M Eifler, and Sarah Northway. On the biology side, Lynne Regan, Alistair McCormick, Ana Casañal, Chris Russo, Judith Debreczeni, and Tristan Croll. An enjoyable environment to work in was provided by the members of the Swain lab: Elco Bakker, Luis Fernando Montaño, Diane Adjavon, and Christos Josephides. And also the Von Delft lab, for hosting me in Oxford and setting up the first CootVR workstation.

For their encouragement I thank my family, Jim, Hilary, Rory, Joel, and Claudia Todd, along with Stephane, Eleanor, Raymond, Rosemary and Rachel Harris. And my best friends, Maya Wedin and Darren Talbot, and fluffy Belinda.

Finally I thank the University of Edinburgh, and the UK taxpayer by way of the BBSRC, for their wonderful support.

Table of contents

1 List of figures

Part I

2 Introduction

Hand-tracked virtual reality is a new technology that changes the way that humans give input and receive output from computers, and therefore may have potential to change the way that I deal with science. It generally allows for a much wider space of inputs to the computer than a conventional mouse, while remaining very “intuitive”. Consequently, VR can allow for new, fast, sophisticated interactions with computer visualizations of complicated phenomena, especially phenomena that are specific to three-dimensional space rather than two-dimensional space (which a mouse is best suited for). However, designing these interactions is non-trivial, because in comparison with mouse-and-keyboard interaction, VR user-interface design is inherently more complicated and currently not well-understood.

Proteins and nucleic acids are an extremely important example of 3D shapes that are of immense importance to our understanding of the natural world. The human body contains thousands of different proteins and there are likely many hundreds of billions more in the natural world. They are also, from the point of view of computer graphics, an unusual sort of shape. Most shapes considered by computer graphics specialists, for example animated characters, are dealt with by programmers and designers as objects with a surface and an interior. But when structural biologists consider proteins and nucleic acids, the conventional view is *ball and stick* - a bramble-bush like 3D shape with no innate definition of “up”. Molecules therefore have to be thought of differently to the shapes of everyday life, and present unique challenges for interaction design.

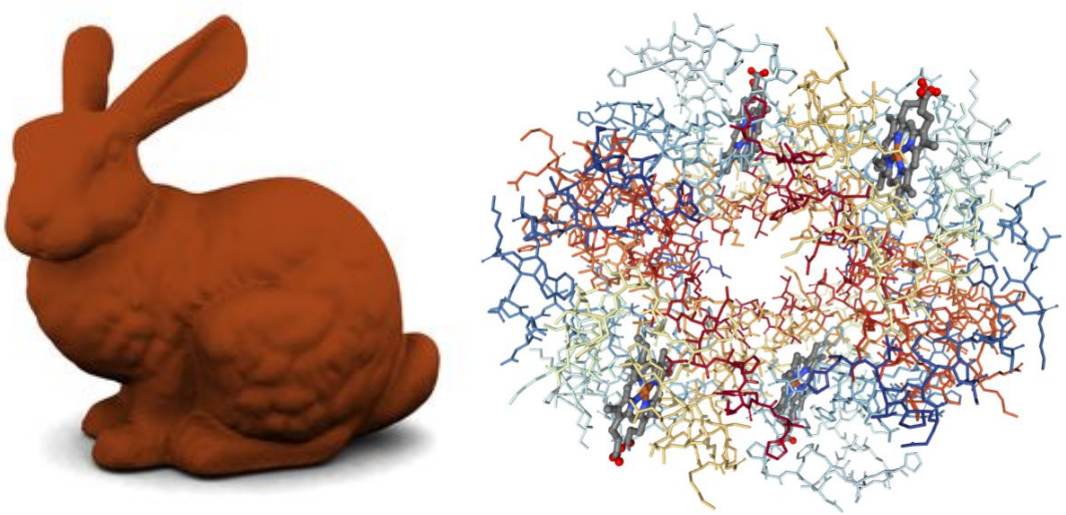


Figure 1: Two 3D models. Left: the Stanford Bunny, a “mesh” used widely as a test case for 3D processing algorithms. It is hollow, and will typically have been be created by a 3D artist who specifies locations for triangles that make it up; it is also an “easy-to-consider” shape, in the sense that a human can have a good intuition about what it will look like if rotated. Right: an

empirically-determined 3D model of haemoglobin[1], a famous example of a biological macromolecule. The details of its interior matters greatly and it is much more challenging to try to rotate in one's head.

The central question of this thesis is: *to what extent, if at all, can VR provide help to structural biologists?* Broader questions that are also of some interest include "to what extent can VR help science or information visualization?". For this project I took up the specific goal of designing, implementing, and testing a piece of VR software called CootVR. The software is intended to be used by structural biologists to manually perform "model building", a task which, since 2002, has in general been performed using a keyboard-and-mouse program called Coot. Originally the goal of the project was to create a wholesale replacement for Coot, but I decided that this would be unwise, for reasons I will describe. But some features of CootVR are enough of an improvement on Coot that I would argue for its use in at least some cases, for example building large loops of protein backbone into 2.5-4 Ångstrom data ab-initio. Additionally, VR technology is expected to undergo advances in the next 5 years that will make it more convenient to use; after this time I would argue that ordinary use of Coot should regularly incorporate CootVR.

Model refinement (which is described in detail in chapter 4) presents a number of complex and interesting problems in the domain of VR user-interface design. It involves modifying three-dimensional models based on a three-dimensional blueprint (which has come from experimental data), while also incorporating existing knowledge of how atoms behave in 3D space. Model refinement is an extremely *visual* task: the user wants to get two three dimensional shapes to "line up" with one another. They do this by changing the shape of the 3D model (i.e. changing the positions of its atoms), raises many design questions that I will address.

CootVR is the first VR program with the stated goal of aiding model refinement. As will be described in the literature review, there have been more than a dozen VR programs that touch on structural biology in some way. Specific proposed applications include docking, education, and drug and protein design. But as I will argue, there are considerable problems with the proposal that VR be used in these domains. The broad problem is that in drug development, docking, and education software, the *manual movement of atoms* is only occasionally useful and it is uncertain whether this will change - I believe that this is the reason that no existing VR program has actually been adopted by biologists. In model refinement though, the entire goal of the visualization is to help the user make decisions about how to move atoms. Since one of the main proposed benefits of VR (by all accounts, including the authors of the reviewed papers) is in the increased articulation given to the user's hands, I believe that working towards this goal is more illuminating as to how VR software design can impact structural biology.

In 2015 when this project began and there were very few software developers with access to hand-enabled VR, there was a great deal of excitement in the community and the feeling "anything could happen" as a result of the way that this generation of consumer VR could change the way humans interact with computers. A general pattern in our project has been that

exotic new approaches to user interface design making use of VR have turned out to be a bad idea, and so I have regressed to design patterns that are not *very* different from mouse-and-keyboard based ones.

3 Review of existing VR software solutions in structural biology

While the exact application I have chosen (model refinement) is very important to us, our primary interest, in a strong sense, is the new hardware platform and how it affects the whole of software for science, especially biochemistry. Here I will examine other software projects that combine VR and structural biology, and argue that model refinement is a currently-neglected opportunity to discover things that are both useful and, from a design point of view, interesting. I will also argue that the virtual reality development community is overoptimistic in its estimation of how VR can change software scientific software in comparison with the mouse and keyboard interface.

3.1 Proposed uses for VR in structural biology

When new computer visualization technologies arise, they are often applied to visualizing proteins. Several dozen projects have attempted to apply VR to structural biology in some form. Various uses have been suggested. I have grouped projects by their proposed use. Some projects[22,34] are simply meant as an exercise for the creator, to prove that proteins can be visualized in VR. As an interesting historical detail, the first project to combine VR and structural biology[56] was in 1992 and fit into this category, and it was performed under the supervision of Jane Richardson, one of the most renowned structural biologists in the world, and Frederick Brooks, one of the most renowned software engineers in the world. I will say no more about these projects, because I believe that a stated purpose can be very helpful in comparing VR with the mouse and keyboard interface and thereby shed light on how VR changes things, and that is our main interest.

None of the projects described below, to our knowledge, have experienced widespread adoption. Some of them, such as Haptimol[7] and Yasara[7,8], are still used sometimes, but only at the institutions where they have been created. I believe that the reason for the lack of adoption of these programs is that they do not solve common or important problems that biologists use 3D visualizations for. Biologists face a large number of problems that take a large amount of their time; the role of software should be to help solve those problems. I believe our work can have a significant impact because the problem it tries to solve (model refinement), *is* a common, important problem.

3.1.1 Education and presentation

Explaining concepts in structural biology and pharmacology can be difficult, and this is in some part because there are concepts that involve fairly sophisticated 3D movements or behaviour which is confusing if described purely verbally. If anything can be done to make it easier to explain a given molecular movement, this can be helpful for biology in a number of ways: it could make the concepts easier to teach to students; it could allow experts to explain conjectures or results to one another at conferences or within companies[9]; and it could help medical professionals explain to their patients how prescribed drugs work (this is an increasingly common way of reassuring a patient).

Figure 2: a chemist uses a plastic model to explain the activity of a molecule (the VX nerve agent)[10]. It is common to twist and deform models like this one in order to convey something.

To explain the concepts, 3D visualizations can be very helpful, and visualizations are traditionally made with physical ball-and-stick models, see figure 2. Computer graphics can be and is used too, for example figure 3. Using computer graphics, it is possible to have a simulation be performed on the atoms, allowing the audience to see how atoms react (for example electrostatic repulsion or the formation of beta-sheets). The program Sculpt, using a mouse interface, allows the user to put atoms in certain states and see how they react[6].

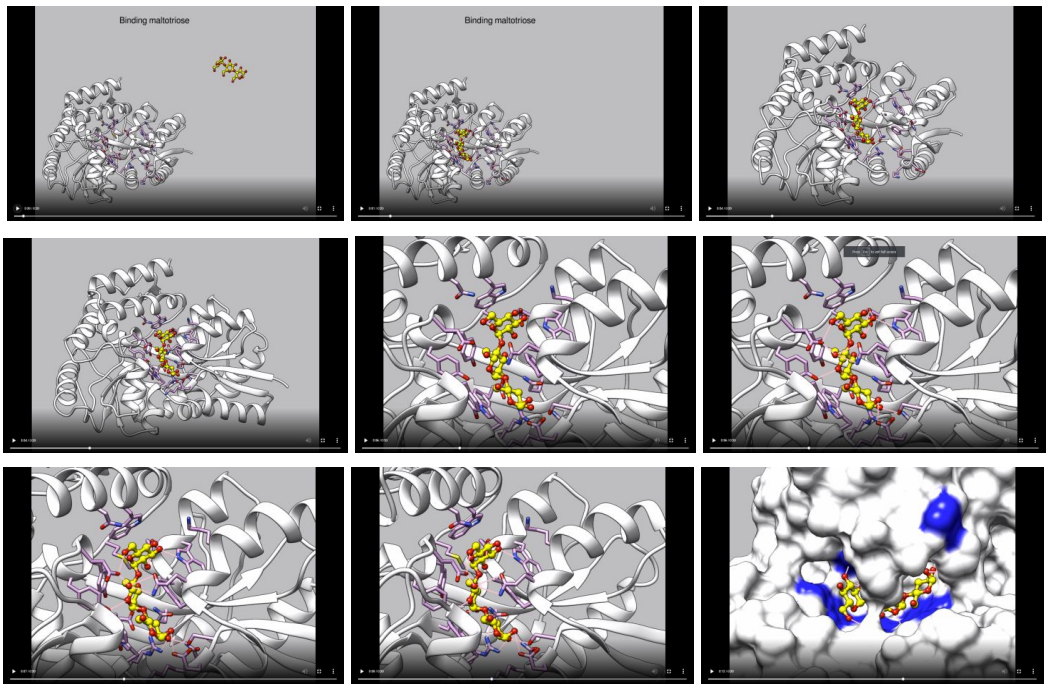


Figure 3: Frames from a video made in ChimeraX[11,12] that displays a sophisticated sequence of 3D actions, with the aim of explaining how maltotriose becomes bound to its binding protein.

There exist multiple pieces of VR software for this[13,14,15,18,14,16], the main argument being that atoms are located in 3D and should be manipulated as such.

Some of the programs are aimed at the secondary school educational sector, others at University education. Of those aimed at secondary school education, many reported that children are able to retain information, and found the content engaging, but that the hardware was difficult to organize[14,15]. A basic form of VR called mobileVR, see below, has been rolled out in some schools and the popular molecular graphics program Chimera allows VR videos of molecules to be played[17]. However, it is not argued that VR videos offer an advantage over 2D videos in this context, other than the fact that children are impressed by VR.

VR as an educational tool is found to be less useful in University education and communication between scientific peers, where it has not been adopted at all, in spite of the attempts above. The reason for this is that the VR experience communicates quite little that cannot be communicated by a 2D video, and is much more difficult to set up, even if it does have a more impressive result. University lecturers, and researchers at conferences, have a large amount of information that they want to convey to their audience, and really only part of it concerns the geometry of molecules. Setting up and donning the headset purely for the purpose of explaining the geometrical part is, in my opinion, unlikely to be worth the trouble in general.

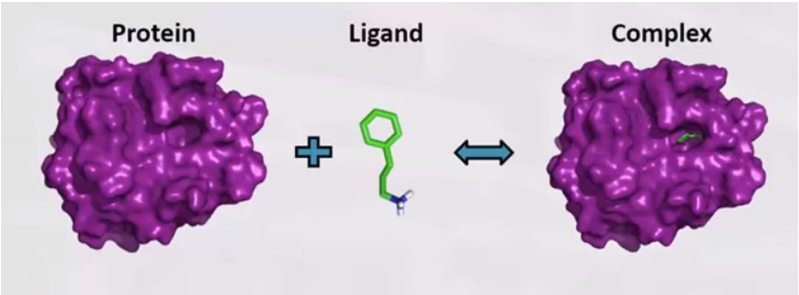


Figure 4: an excerpt from a powerpoint presentation on drug docking. It is tremendously simple, conveying quite little geometrical information, such as how the torsion angles are changing. And yet it successfully communicates all the information that its creator cares to communicate - no extra perceived value is given by donning a headset in order to show the geometrical details of the dock.

Figure 5: an early “mixed reality” video that I created, which talks about the hepatitis vaccine. It can be seen at [18].

3.1.2 Protein-protein docking

“Protein docking”, in the context of *computational* biology, means working out how two proteins “fit together”. Proteins do this within cells for various reasons, and understanding the geometrical details of how a given pair of proteins do it is very worthwhile, because it can, for example, suggest where the proteins might be mutated in order to prevent them from docking.

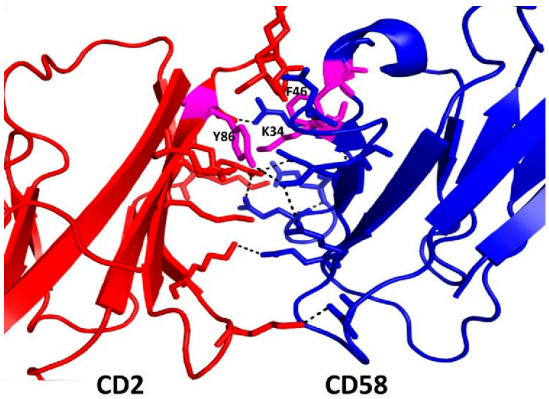


Figure 6: a pair of proteins, “docked” together[19]. This configuration for them will be “stable” for some time.

There are many pieces of software for protein docking, the most famous one being HADDOCK[20]. They take as input a pair of protein structures (PDB files, described below) and output the position and orientation of one relative to the other with the optimal docking situation. This must be measured according to some understanding of how atoms interact; this is formalized in the concept of a “forcefield”. In the context of docking, all that the forcefield has to say is that, for example, two atoms overlapping is “unfavourable” whereas two hydrogen atoms placed at the right distance for a hydrogen bond to form is “favourable”.

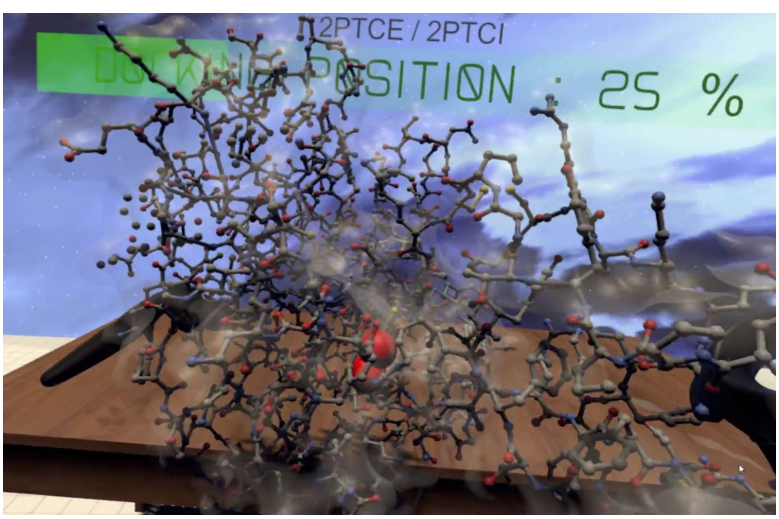


Figure 7: *Bioblox*, a VR protein docking program. The two black objects on either side are the user’s hands, and are manipulating a pair of molecules.[21]

VR protein docking suggests a very obvious user interface: the user holds one protein in one hand, and examines the interface as they put them together. The user can, while performing the docking, be informed as to the quality of the dock according to the forcefield, potentially using auditory feedback. This is how all VR protein docking programs work[7,22,23].

Many of these projects had disappointing results, one of them finding that human-produced docking solutions were always worse than computer ones[24,25]. Some authors were more optimistic, finding that users could perform well in that they were sometimes able to replicate experimental findings[26,24,27], which is the standard of assessment of theoretical docking methods. Although the authors still concluded that after a human has found a basic fit, “the refinement should be carried out by the computer, which is much faster and more systematic”[22].

There are many fundamental problems with this method which I believe explain why no VR protein docking program has been widely adopted. For a biologist to decide whether to use the VR-based solution, they need to ask what advantage it confers in comparison with existing solutions. The best thing that can be said in favour of the VR solution is that it can, in theory, save time - the way that non-VR docking works is a variation on “brute force”, i.e. every possible conformation is sampled, which takes a large amount of computing time, maybe several hours or days.

This argument does not work in practice though. The time-saving must be weighed up against the time that is lost from setting up the program and hardware, which is considerable; having a computer run for several hours is not much of a problem if it saves a user an hour for themselves. The suggestion also fits into structural biology culture in an awkward way - protein-protein docking is, for most people, a task that needs to be done only once in a couple of years; it is not very worthwhile to optimize such a process. If there is any extent to which the programs requires developing a skill in order to operate them, then this costs even more time.

It is worth noting that manual protein-protein docking is not something that has been implemented in any of the many non-VR programs used for protein docking. This would suggest that if docking had been widely adopted in VR, it would been a capability *specifically enabled by VR and never before possible*. This would have been very persuasive in proving VR’s potential, but is probably “too good to be true”, by which I mean that the fact that nobody has attempted to implement a protein-docking interface without VR implies that the structural biologists who actually use the results of docking programs are comfortable letting them do it entirely automatically.

3.1.3 Rational drug discovery and protein design

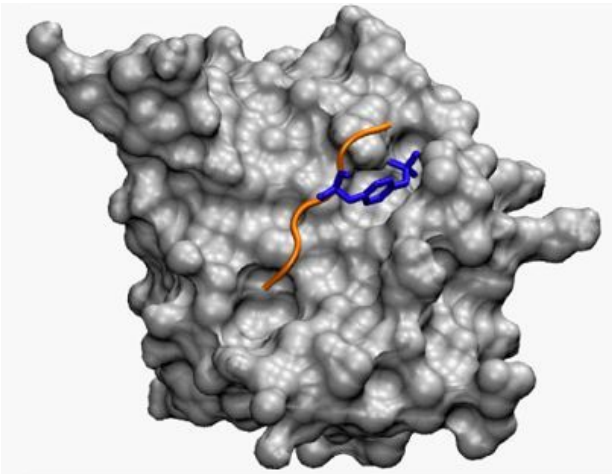


Figure 8: drug-protein complex, with the drug in blue and visualized using a different method from the protein[28]. The specific shape of the drug has been “designed” specifically for the protein.

Rational drug discovery is a field in which conjectures are made as to what drug molecules may help cure diseases, and therefore has enormous importance for global healthcare[29]. The most widely-used theoretical approaches to rational drug design involve taking a protein structure (a PDB file, see below) and attempting to find a molecule that will bind to a certain part of it the protein, with the hope that it will change the protein’s behaviour in some detectable way if the drug is synthesized and administered in vitro. When I say “find” a molecule, I mean making a molecule with a particular connectivity, usually out of less than thirty common atoms. Protein design is similar, except that the molecule being designed is a protein, and so is large, but is under a particular set of constraints.

The argument for VR in this domain is that the molecules are again 3D and therefore merit being examined in 3D. For example, most non-VR molecule visualizations hide the hydrogen atoms to improve visual clarity; in VR this is unnecessary, providing an advantage since hydrogen atoms are important for predicting bonding. There are a large number of programs in this domain[24,13,8 22,25,9]. Some projects have involved synthesizing molecules made in the program[9]. But again none have been widely adopted; the abundance of people attempting to do it is, I conjecture, best explained by the fact that it is an endeavour which, if it were ever to be at all successful, would be very lucrative.

One problem with this approach comes from the experimental economics of drug design/discovery. There is a particular currently-accepted way of making decisions about what atoms should be placed at what positions in a drug. This method is to do a huge number (potentially tens of thousands) of experiments, using robotics, with each one using a different drug, each differing from other drugs by only a few atoms[30]. The cost of screening all of these

variants is only slightly higher than screening a single variant. This means that there is not a great deal of purpose in proposing a single specific molecule. It may certainly be worthwhile to propose a space of variations on a single molecule, but it is much harder to argue that this is something human intuition and 3D visualization is well-suited to.

3.1.4 Model refinement

I will give a precise definition of model refinement in a subsequent chapter. For now it suffices to say that I believe that model refinement is better than any other modelling task as a testing-ground for the usefulness of VR. The main reason for this is that it is an example of an already-existing visualization-based problem where users are regularly required to move atoms manually.

Some model-refinement programs are automatic, others are manual and involve interacting with a visualization. Speeding up automatic procedures for model building and refinement is worthwhile and a major goal of computational structural biology. However, the larger drain on structural biologist worker-hours is the manual part of model refinement, which can take months - even a 10% or even 5% speedup offered by VR would be welcomed by the community.

Currently-existing non-VR-based programs for model refinement include Coot[31,32], O[33], Isolde (when it is not combined with ChimeraX)[34] and Molprobity[31]. All of these programs are highly specialized for the task.

There is one VR program that has been discussed in the context of manual model refinement, which is ChimeraX[35]. ChimeraX's design has not exactly been optimized for model refinement though - essentially its interface in VR is simply taken from the desktop interface, making it very hard to use. The paper reporting its consideration for model refinement concluded that it would currently be more of a time waster than a saver, and that this feature of it is not a development priority.

There are two other VR programs that can theoretically be considered for model refinement, because they have an "energy minimization" functions and the ability to move atoms. These programs are Foldit[36] and Sculpt[6]. But neither of these are exactly intended for model refinement though, among other things because they do not include visualizations of electron density.

At one point support for the Nintendo Wii was considered for Coot[37]. This would represent a significant change to its control scheme, but the Wii lacks the functionality to really be considered VR.

3.2 General criticisms

There are some general criticisms that I have of many papers from within the literature bridging structural biology and VR. Early on in this project I made some of them ourselves, but after this I specifically aimed to avoid them. It is by no means the case that all authors make these mistakes, and certainly not all to the same extent. Nevertheless it is worth talking about, as it contributes to the background of this project.

3.2.1 “Coolness” as a goal

Figure 9: a still frame from the famous 1993 movie “Jurassic park”, during a scene where a character is walking through a laboratory that is supposedly doing cutting-edge biological research. A character wearing a headset is holding a model of DNA and rotating it. The setup is included in order to give an understandable demonstration to the audience (laypeople) that the lab is “cool” and “futuristic”. Whether this method of looking at DNA using VR is a good idea would not at all have factored into their decision to create the DNA-VR setup. A major concern for VR research is whether it is sufficiently motivated by the same thing.

Many of the studies cited above talk about how much the user “enjoyed” the experience of using the software, or found it to be “cool”[21,38]. But a piece of software can be very cool without being at all useful. Even if the positive feelings that a scientist feels towards a program for its coolness last for a long time (this seems unlikely), their goal with their work should be to make scientific discoveries, and if there is an alternative piece of software that can help them with their work in a way that is 30% faster, but less cool, then they are more likely to use the less cool option.

I am not saying that because something is cool that this necessarily means that it *isn’t* useful or efficient. I am simply saying that coolness is orthogonal to usefulness. *Other things being equal*, it is good if a piece of software is cool or enjoyable. However, at this stage in the development of VR as a software platform, it is not at all clear that “other things are equal”, because VR is not being directly measured against mouse-and-keyboard solutions; that is something I aim to

contribute to. Additionally, there are at least some ways in which aiming to make something “cool” can cause it to become less useful (again other things being equal), such as wanting to make something “more interactive”, and therefore less automatic, and therefore more time-consuming to use.

In this thesis, I will describe many software features that I implemented, and it should be admitted that some of them were originally implemented with the motivation that they might be cool. However, I aimed to be objective and would also implement things if they appeared less cool than what I already had.

The argument above mostly also applies to the term “natural”.

3.2.2 Misunderstanding of the capabilities and goal of current molecular dynamics techniques

Several of the projects described above across the education, docking, and drug design applications, involve the user interacting with a simulation of a molecule, i.e. a visual representation of it such that its atoms are moving in ways that are meant to imitate the way that they move in the real world, which is called “molecular dynamics”. The claim is sometimes made that users may obtain scientific insight by watching this happen, and also by interacting with the molecule in 3D, eg purposefully destroying the fold of the protein in places and seeing how the molecule as a whole reacts[39,40,14,41].

Figure 10: proposal for how chemistry research could be conducted in future, from the “Dynamicland” whitepaper[42]

There are a number of problems with this idea. One is computer power - calculating the trajectory of a set of bonded atoms can take days on a powerful computer, whereas in order to crete the illusion of movement and responsiveness(“real time”), computer graphics require updates around 15 times per second[37,43]. This problem is well-acknowledged in the papers

above; they tend to state that their goal is to research and develop systems that may be useful once this problem is solved; this is reasonable.

But even if this problem were to be solved at some point in the future, other would remain:

1. Chemical activity of interest to biochemists happens over a timescale measured in microseconds. But for atoms to move at a reasonable speed for a human being to watch and understand, they have to move somewhere around a nanosecond per second. This is to say that to watch (or interact with) a molecular dynamics simulation showing even the most basic biochemical activity would require either seeing it as a jittering blur, or to watch it for hours. The latter option is time consuming; the former option would make it very hard to glean anything from what one was seeing.[44]
2. Biochemists are interested in the statistical behaviour of molecules, i.e. "what they do on average", rather than what they did over the course of one *particular* simulation. Consequently a single simulation is not very useful, because it is not known whether the series of events that occurred was a typical one. To visualize or manipulate a statistically-informative simulation, it is not sufficient to show atoms as balls at particular locations.
3. When "playing with a simulation" such as a video game, human beings are naturally compelled to interact with it a great deal (this is touted by several papers as an advantage). But in the real world, atoms do not have conscious beings moving them around, so for a user to assume that atoms behave in the way they saw them behave would be misleading. To put it another way, the purpose of molecular dynamics simulations is really to "see what the atoms do on their own"; the only place where human intervention should be allowed is in setting up "initial conditions" for the simulation, but when this is done, care is taken to ensure that bias is not introduced.

Molecular dynamics simulations are now a fairly widespread way of investigating molecules. When writing papers, in general the community is happy to simply supply or received statistical information and possibly a video, see figure 11. Note that the video, while being useful, is only meant to supplement the statistical observation, which might be something like "the event seen in the video has a half-life of 10 microseconds". For this illustrative purpose, it is better for the authors to provide a video than it is to provide an interactive version of the simulation.

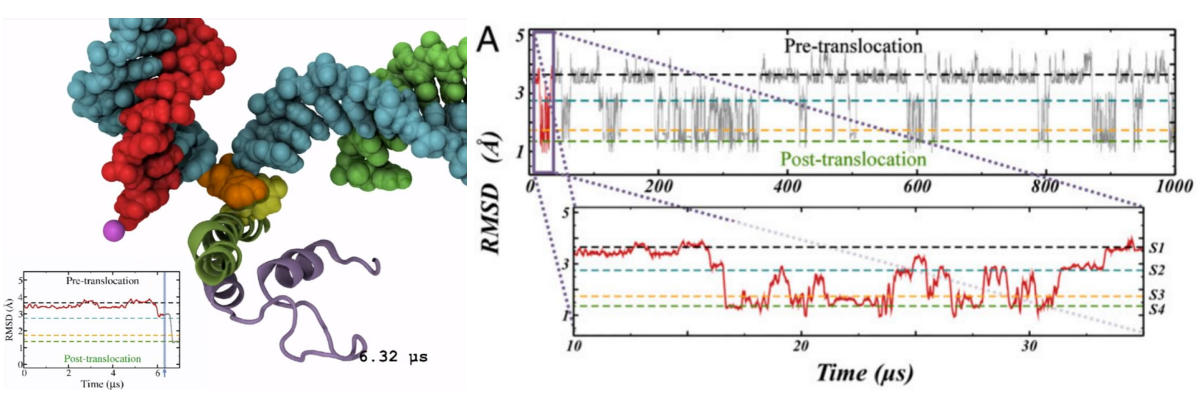


Figure 11: The output, as given in a paper with a supplementary video, of a molecular dynamics simulation of an RNA polymerase II translocation[45]

In general, biological sciences, including structural biology, are at a stage where theoretical models cannot be trusted very far - as described above the fact that it is still more economical to experimentally assess many thousands of potential drugs rather than assess them on a computer is an example. The concern that takes up time for biologists is therefore the interpretation of data. I consider it a pity, therefore, that VR researchers looking into biology have concerned themselves so much with simulations. One application of VR that I considered looking into but did not have time for was the segmenting of cryo-electron tomography data - this has never been attempted, even though I would say that it would seem a fairly obvious candidate for an application of VR.

In some cases, for example [46], the reason for the focus on theory rather than interpretation of data is cultural, with VR researchers being in computer science departments rather than biology departments, potentially with them being told the basics of biology (enough for them to imagine erratically-moving molecules) but not having familiarity with the needs of biological research.

3.2.3 Overestimation of the power of human intuition

It is common within discourse around VR to argue that it can expand what human intuition is capable of. Some characteristic quotations are “Humans are naturally skilled at visual pattern recognition in a manner not (as of yet) replicable by computer algorithms, and engaging this skill is a potential tool for speeding up automatic searching”[47] or “human brains are 3D computers”[48]. A reasonable interpretation of these (often quite delphic) statements in the context of structural biology is that VR can be used by humans to train ourselves in how atoms react to one another, and if I do this enough then I can answer major questions in chemistry using the intuition I develop[24].

I would say that this argument has been taken too far. To answer a given scientific question, human intuition is a reasonable starting point only if it is later formalized, quantified, and argued for. Instead of using intuition to answer a question, scientists usually have two other options: *formally deriving* a guess based on previous experiments; or doing an actual experiment. Faced with a decision about which of these to use, ideally a scientifically-informed economic calculation should be done, where the main factors are accuracy of results and time investment by human beings.

Consider the example of docking. In terms of accuracy, humans do not do well, even with VR: it has been said that human 3D intuition can allow us to do better than brute force[24], but the first study into it concluded, in a sense, precisely the opposite: that when a human is attempting docking then they should adopt a time-consuming brute-force like approach[25]. In terms of time investment, intuition does badly too - the canonical example given of the power of 3D intuition is the results of the “foldit” project[14,47]. However, the only example of a major problem solved using

foldit involved so much human effort that it is likely that it could have been solved using brute force[14,47,49].

The ability of VR to “tap into” human intuition is sometimes also exaggerated. When working with our hands to do tasks like tying shoelaces, I rely on our sense of touch to guide what is being doing, and this is absent in VR - even though in promotional demonstrations it is sometimes made to seem as though the sense of touch is present.

4 Model building and refinement

“Model building” is a well-defined, and important, task within structural biology. Formally, it means making an accurate list of atoms that exist in a protein or nucleic acid, specifying their elements and coordinates (a “model”). This (generally) first requires an experiment involving the molecule which acquires an “electron density map” for it. Decisions about where the atoms are likely to be located can then be made based on the values in this “map”, essentially because at places where atoms are located, electron density values can be expected to be higher.

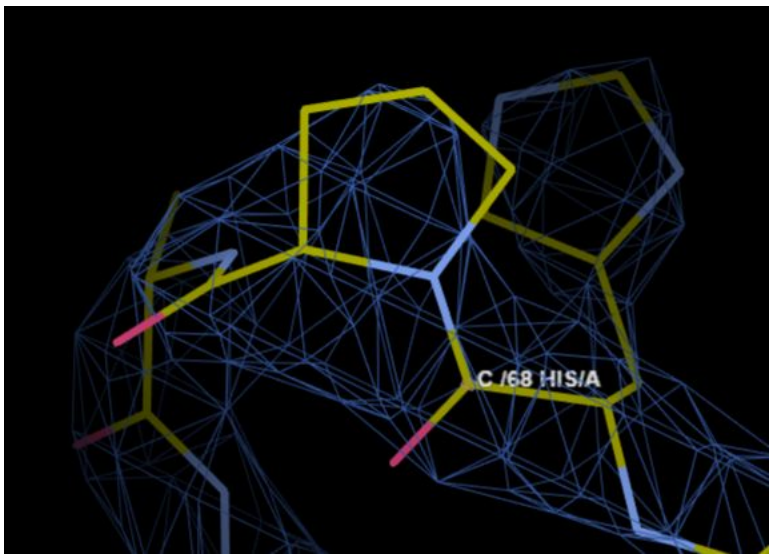


Figure 12: “model” and “map” visualized together

It is rare (strictly speaking impossible) to obtain the “exact” electron density for a molecule. The level of detail in a given electron density map is referred to as the “resolution” of the that map. It is quite directly analogous to the resolution of a photograph. Resolution is measured in Ångstroms, denoted by Å, one Å being 10^{-10}m. In electron density data that is said to be 0.5Å, one will have measurements of electron density that are spaced in a grid 0.5Å apart.

Knowing the positions and elements of atoms in a protein is the general goal of structural biology, and is tremendously useful to biology as a whole, because it illuminates how the biomolecules behave and function. For example, before 1956 it was generally known to biologists that DNA stored information that could be passed on from parents to children, but it was not known how. With x-ray diffraction and a famously arduous process of model-building, it was determined that DNA’s structure was a double helix, see figure 13, with adenine pairing with cytosine and thymine with guanine. Questions still existed, but figuring out how other proteins could manage a double-helix shape became a clear way to make progress on any biological question.

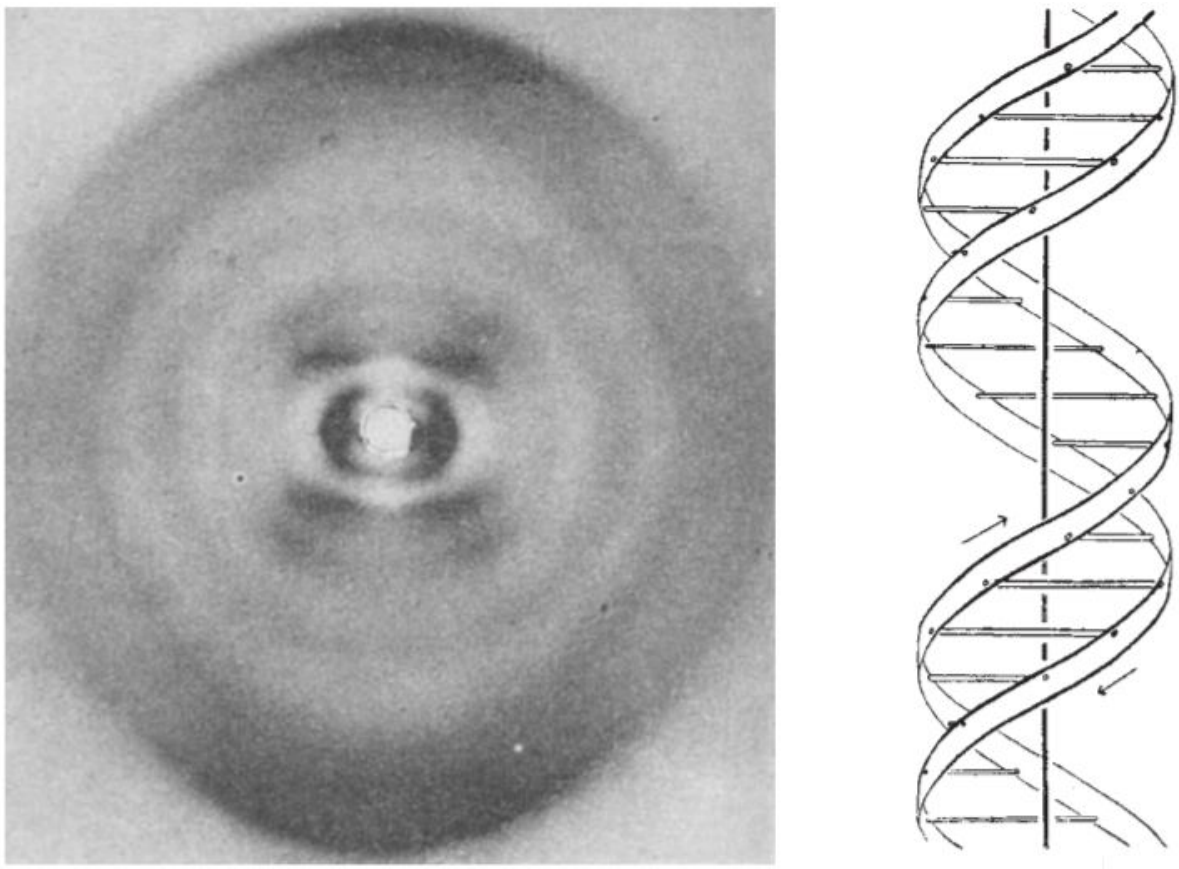


Figure 13: a fibre diffraction pattern, and the proposed model explaining it. From the original publication describing the structure of the DNA double-helix, generally agreed to be one of the most important scientific discoveries in history[50]

Another example of the importance of structural biology, and of creating properly-refined models of proteins, is its use in the pharmaceutical industry. Large parts of pharmaceutical research, so called “rational drug design” often boils down to a geometrically quite simple task: one wants to make a small molecule which will attach to a specific protein in a way that causes the protein to behave differently. A simple example of structural biology being pertinent to a somtimes-encountered situation in real-world drug design is: there is some 11-atom molecule X that is known to bind to some protein A and decrease its activity, which prevents a tumour from growing. However, molecule X also binds to protein B, which prevents the liver from working properly. In this situation it would be desirable to create a molecule Y that binds to protein A but does not bind to protein B. In order to create molecule Y, it is worthwhile to make a visualization of how molecule X is interacting with both protein A and protein B; see figure 14.

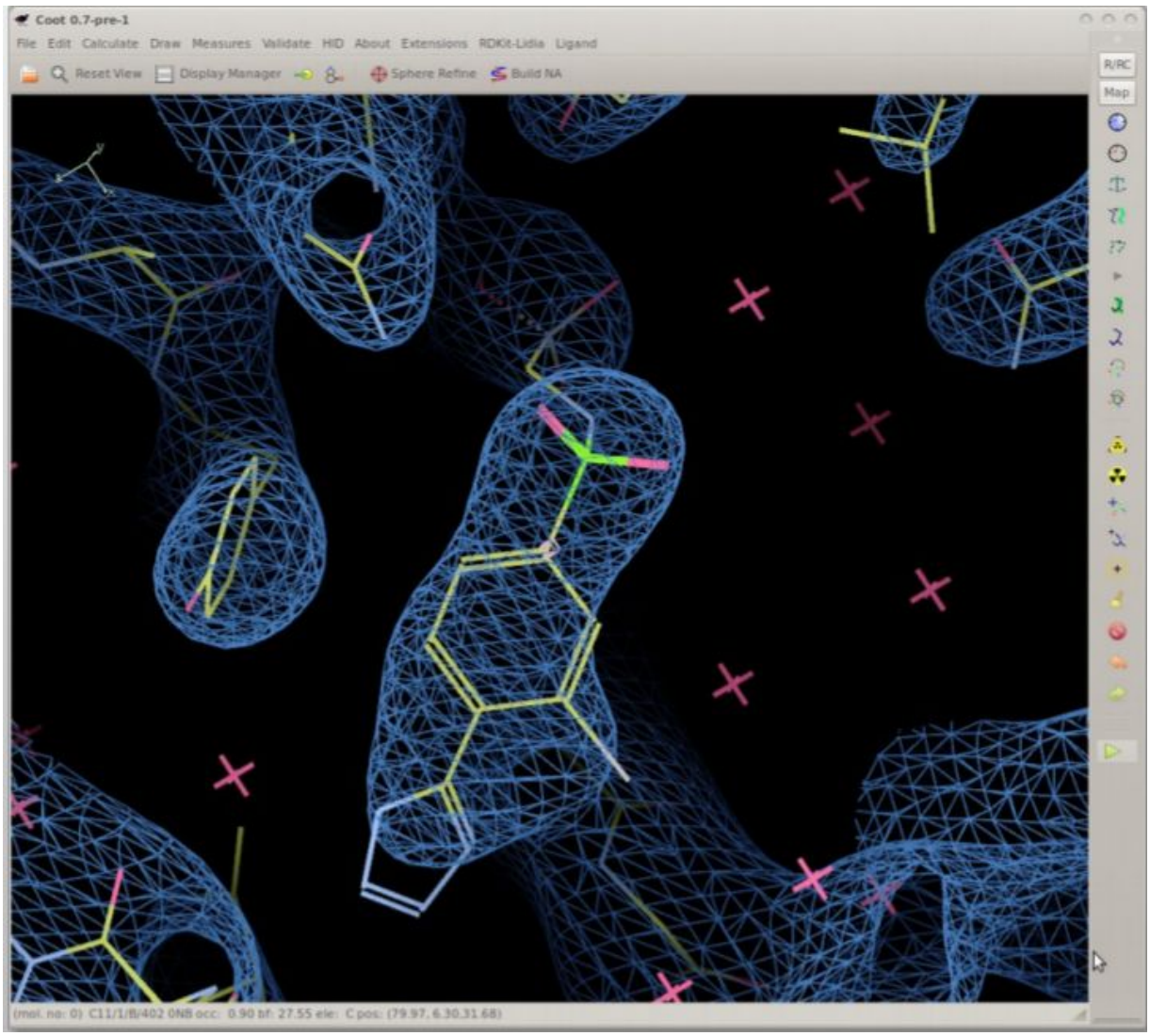


Figure 14: An example of Coot being applied in drug design. The molecule in the center of the screen is a drug. It is held in place by the atoms that surround it. If one were to make a variant of this drug which differed by one atom, there is a high probability (but not a certainty) that it would no longer be held in place. From [51].

4.1 Data sources

There are two kinds of experiment that structural biologists use to obtain electron density data ("maps"): **electron cryo-microscopy** and **x-ray crystallography** (also known as "crystallography")[1]. Both of have advantages and disadvantages. Prior to model refinement, the data from these experiments is processed in different ways, both having implications for our

[1] Cryo-electron *tomography* has been used too, but this is quite controversial and not discussed in this thesis.

project. Some of the differences are summarized in the table below. In the following sections I will go into more detail on them.

Crystallography advantages	**Electron microscopy advantages**
Higher resolution No “orientation bias” effect No lower limit on the size of molecules that can be worked with	Capable of working with larger molecules Does not require crystallization (and so can be less time consuming) Higher probability of success

Table: Crystallography compared with Cryo-EM

4.1.1 Single particle cryogenic electron microscopy

Here I will describe single-particle cryogenic electron microscopy and refer to it as “cryo-EM”; other kinds of electron microscopy are not of interest, as they are not able to get particularly high resolution. Cryo-EM sample preparation involves purifying the sample protein, putting multiple particles that are copies of it on a grid, rapidly cooling it to an extremely low temperature, and putting it into an electron microscope.

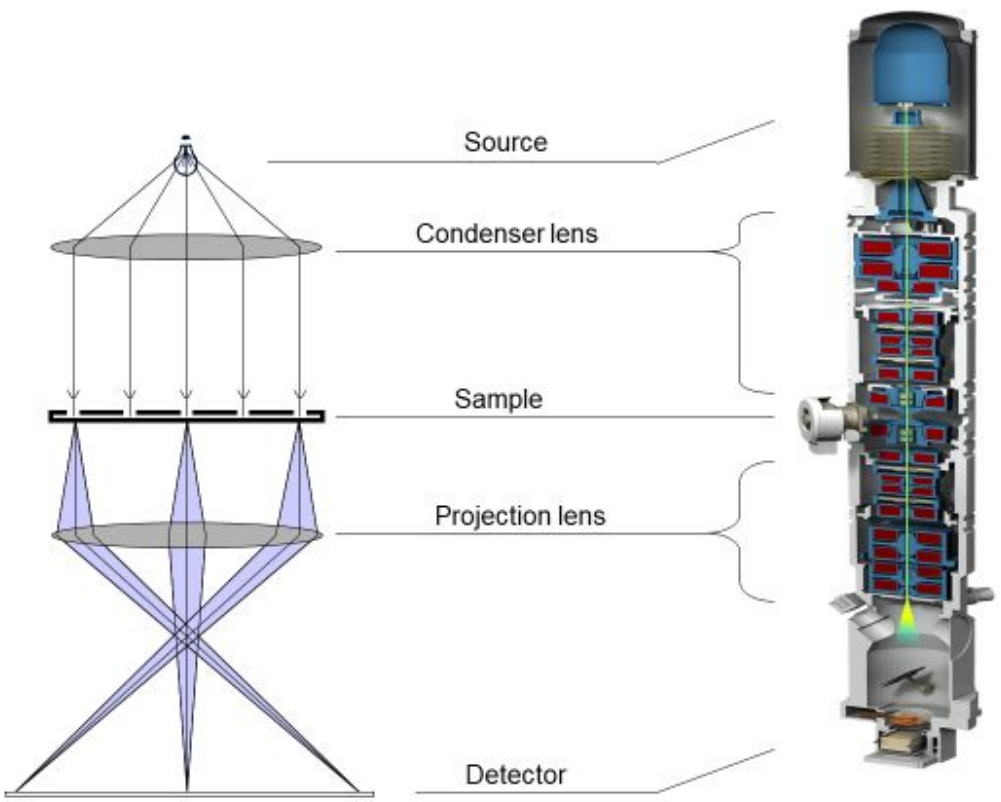


Figure 15: cryo-electron microscope, from [52]

Inside the microscope, electrons are fired at the sample, allowing and an image of its 2D projection is captured; the plate is then rotated and this happens again. There are multiple copies of the same protein inside the microscope, so thousands or millions of pictures of it at different angles can be captured, see figure 16. These images can then be put in the same place in 3D space and aligned to get an electron density map[2].

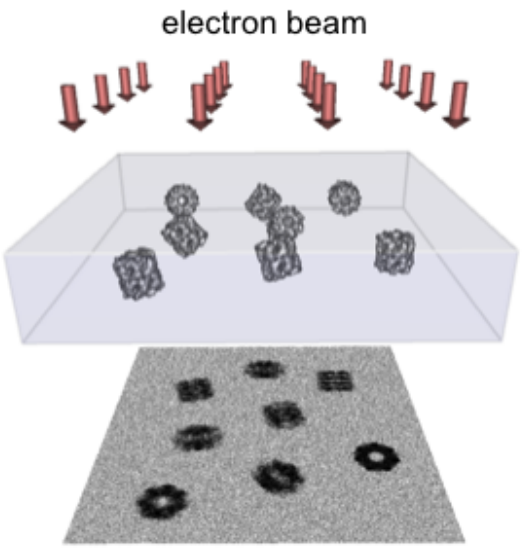


Figure 16: An illustration of particles in vitreous ice, having images of them captured that lead to cryo-EM models [53]

One problem is that, sitting on the plate, there can be certain orientations that the particles are more likely to be found at due to their shape, so not every perspective on the particle can be captured, or at least the data from some perspectives is noisy, due to not having a large sample size for that perspective. Consequently some regions of the particle can have much higher resolution than other regions.

[2] Since this is a 3D movement task, I have inquired about whether it is worthwhile to try to do this in virtual reality, but have been advised that the dataset size is too large.

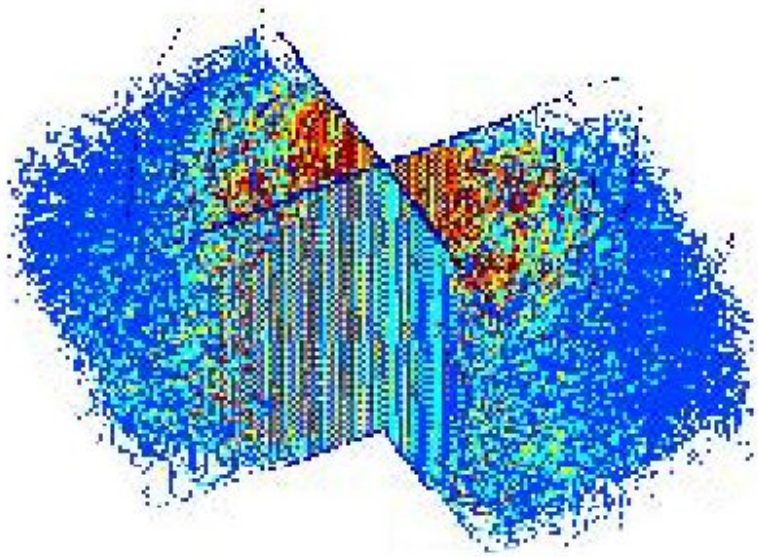


Figure 17: The missing wedge effect[52,54]. Points represent locations in a 3D volume that have a reading. Blue points have few readings, red have many. This example is particularly bad; the missing area usually has at least a few readings to go on.

Cryo-EM data suffers from lower resolution than crystallography data, also in general due to sample size. However, its resolution has increased drastically in the last two decades (the “resolution revolution”), and it is believed that it will continue to get better[55].

4.1.2 X-ray Crystallography

Crystallographic/x-ray diffraction data on proteins is the main way that structural biology has historically progressed. In comparison with cryo-electron microscopy it yields results that are (currently) much higher-resolution, but it can also be much more difficult, depending on the size of the macromolecule. If the macromolecule is too big, it becomes essentially impossible, and cryo-EM is the only option.

Sample preparation involves purifying the protein, and then crystallizing it, i.e., creating a crystal that consists of many millions of copies of the protein, stacked in a regular pattern. This crystal is then cooled and put into an apparatus that, like cryo-EM, can rotate it to many angles. X-rays are fired at the crystal, which diffract off it, creating images on the plate behind it, which are captured.

The diffracted images consist of a large number of diffraction peaks. Their positions give the space group of the crystal, and their intensities give information on the shape of the density map. The information is incomplete however. The intensity of the peaks gives only (after receiving a constant multiple and a square root) the amplitudes of *structure factors*, which are the fourier transform of the electron density map. For the map to be determined, the phases of the structure factors must be known.

Since it is simple to determine phases if the map is already known, a model for the map is assembled. Molecular replacement is the most common way to do this, but not the only one, with previous structures being solved with experimental phasing. An iterative process is then employed: a set of atoms and positions is “proposed”, from which phases for the structure factors can be inferred with a fourier transform. These phases are unlikely to be accurate at first, but every minor improvement allows the map to be recalculated (every atom makes a small contribution to every structure factor) and have the atoms moved appropriately, either manually using Coot or automatically using Refmac[52,54,56].

4.1.3 Sequence data

A very important aspect of a protein or nucleic acid is its sequence, this is the list of small molecules that are chained together to make it up: amino acids in the case of a protein and nucleotides in the case of nucleic acids. Each amino acid has around a dozen atoms connected in a well-understood way, and is connected to the neighbouring amino acids in a similarly well-understood way. Knowing the sequence is therefore a profoundly useful for structure determination: instead of figuring out what one is looking at, model refinement instead can be about figuring out how to arrange a known set of specific things. Practically speaking, sequence data fully determines the *elements* of the atoms one is looking at, so this will not be discussed much in the below.

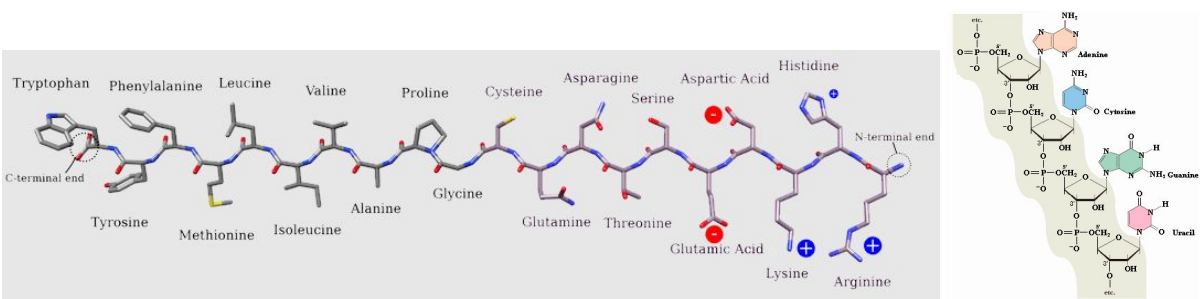


Figure 18: the general shape of the two fundamental objects of interest in structural biology, an amino acid chain, left, from [57], and a nucleotide chain, right, from [58]. Both of these objects are a series of known chemicals that are chained together.

There are several ways that sequence data can be deduced, for example mass spectroscopy. I will not consider any further details in this thesis, and will view protein sequencing as a completely reliable black box.

It is possible to try to build a model while ignoring the sequence, generally with the intention of adding the sequence later. The general approach is to assume that the entire chain is made of one repeated amino acid, usually alanine; if the sequence is determined later, it is, in principle, not hard to change the amino acids after the fact. Ideally this can be done, but if the sequence is not forthcoming it is not necessarily a bad thing.

4.2 Output

It is worth understanding the cultural change brought about by the introduction of cryo-EM alongside crystallography. Again crystal data has high resolution; it almost always allows us to determine the positions of individual atoms. It was also developed before cryo-EM, and so much discourse in structural biology built up around it; most notably, the community adopted the "PDB" file as the de-facto standard, and many algorithms were built using it as input. A PDB file is essentially a list of atoms, with optional additional information such as what residue each atom is in, what alpha-helices exist, etc, but the atom positions are the fundamentally useful thing.

Cryo-EM should in general be thought of somewhat differently. A structural biologist using it and obtaining, say, 3.4Å resolution data might like to use some of the algorithms developed for analysing structural data. But while 3.4Å is, loosely speaking, enough to get some idea of a structure, it is not enough to know the positions of all the atoms. But the biologist still has to make a PDB file, so it can be the case that conjecture plays a part in atom positions. Confidence levels can be given for the positions of each atom, often with alpha carbons having higher confidence than other atoms, especially those in the side chains (this will be of relevance below, where I argue for the utility of making temporary models where alpha carbon locations are proposed but the positions of other atoms are not closely considered). It is then *hoped* that anyone who subsequently inspects the PDB file (perhaps downloading it from the Protein Data Bank) appreciates that the atom positions are unreliable, ideally checking exactly how reliable each is.

4.3 Refinement constraints and automation

A good model fits the data in a way that is *physically plausible*, which means that the model is probable given current scientific understanding of the local behaviour of atoms. The purpose of much discourse in structure determination is to work out how to make such models, and the entire purpose of Coot is to give an interface to our understanding of the laws of physics such that it is easy to create models that conform to them and also reflect the data.

I will give an extreme example of the importance physical plausibility. If the electron density data *appears to show* two atoms that are *extremely* close to one another, say at a distance of 0.1Å, the data might be compelling as to put them that close. However, the scientific community's best understanding of the laws of physics would suggest that two atoms placed within 0.1Å would start a nuclear explosion. Clearly this cannot have happened in the lab, so our data appears to be in contradiction of our best understanding of the laws of physics. In this situation, the laws of physics "win". By this it is meant that the conclusion will be that *the dataset is faulty with regard to the position of these two atoms*; probably it will be double-checked whatever processing it has been subject to to make it show atoms so close together, and there will be a high level of

confidence that a mistake has been made at some point in that processing. All of this is to say that even the best dataset is not the “final word” on the shape of the molecule.

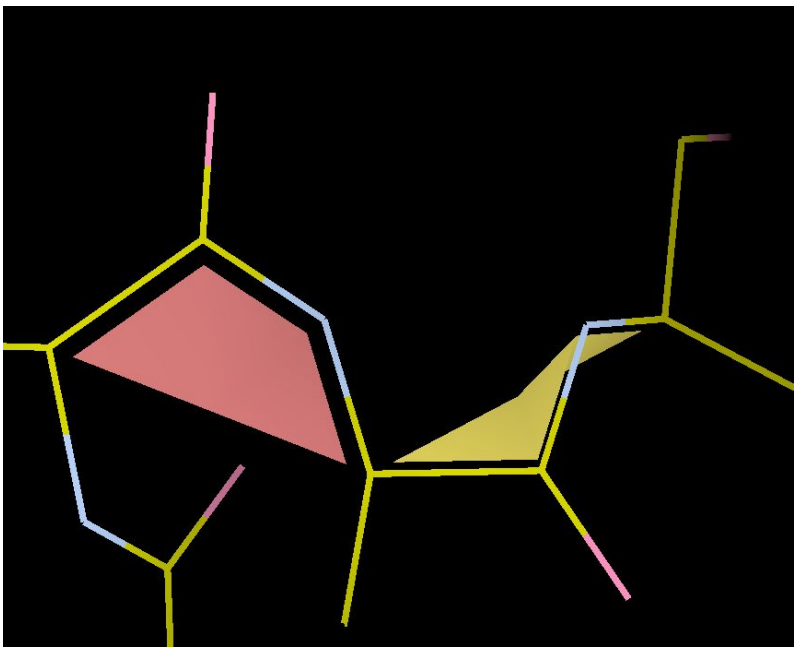


Figure 19: a visualization of a problematic piece of a model, visualized in Coot. All of the atoms in this image are the correct distances from one another and the angles between bonds are all appropriate, however there is still a “peptide omega distortion” in the amides.

To give a subtler example from an actual model, figure 19 shows two amide groups. Amide groups are *expected* to have all of the atoms in them be in a flat plane - i.e. the red and yellow shapes should be completely flat but are not. In all of the very highest-resolution electron density maps, amide groups are planar, and there is no reason to believe that things should be different in lower-resolution maps. This is again true based on our best understanding of physics. *Some* deviation from the plane may be expected, but very little[59]. Therefore, a simple check for model quality can be implemented: if an atom in the amide is outside of the plane by more than a certain amount, then the model is wrong and the atoms ought to be moved from their current positions - even they appear to line up well with the electron density data.

Our confidence level about atom positions can be high enough that I can search through our models looking for “deviations” from our expectations and use them as “red flags” for things that may need to be checked over again.

The example above, in the context of Coot, is usually called “peptide omega distortion”. There are several dozen other metrics that are used to assess the plausibility of the positions of small sets of bonded atoms within biological macromolecules, also backed up by experimental observations from high-quality maps. One which will be discussed later is the *ramachandran score*.

An example of a very useful dataset that is informative as to atomic interactions is *rotamer libraries* (also discussed below regarding Coot-CootVR interaction). If a protein is thought of as being divided into *mainchain* and *sidechain*, a *rotamer* in this context means a set of locations for atoms in a single sidechain. Rotamer libraries contain, for every different amino acid, a set of several possible rotamers, which again are based on high-quality data. Because of them there is very little need to manually move atoms, at least not within a sidechain. Their existence, success, and adoption in the community is a major reason why virtual reality may not be so useful for model refinement as might be expected, because they rule out the need for much manual adjustment of atoms.

4.3.1 The Isolde approach

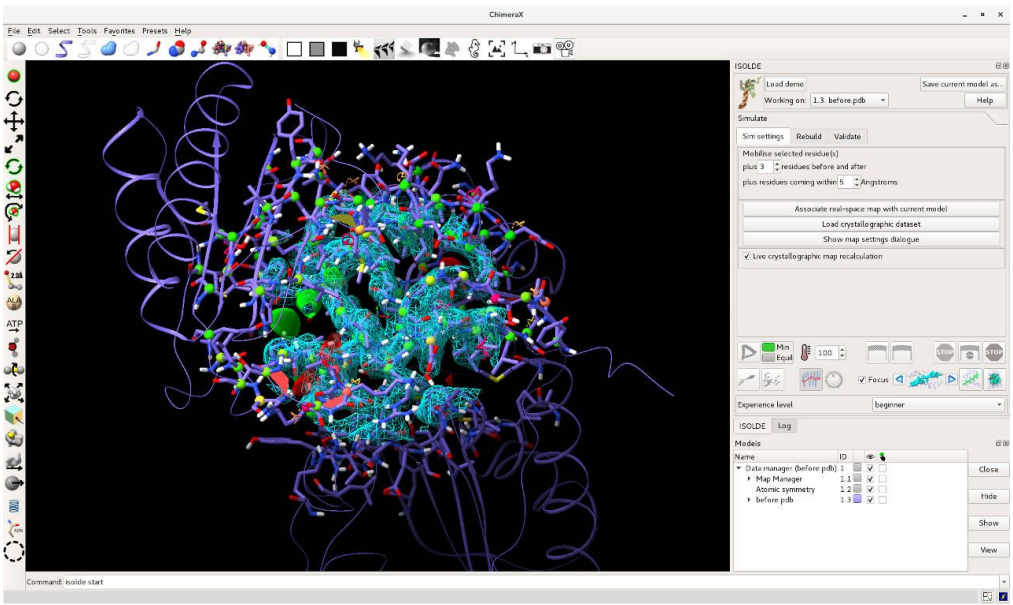


Figure 20: user interface of “Isolde”, a program with the same purpose as Coot

Coot’s approach is different to the one used by the model refinement program “Isolde”, figure 20. The Coot approach in general, which rotamer libraries are an example of, is to use experimentally-derived observations such as “oxygen-carbon bond lengths should be in the range 116-143pm”, or “atoms in an amide should not be further than 0.1Å from the amide plane”. But these can all be thought of as heuristic approximations or interpretations of a “deeper” set of facts, specifically the system studied in the field of quantum chemistry. Isolde’s approach is to use molecular mechanics, described above, to make it so that the atoms are always moving

The simulation approach offers several advantages:

1. One can, superficially, see “clashes” between heuristics - for example where one would an atom should be move left, another would say it should be moved right. In this

situation, a resolution always exists so long as the heuristics are reliable, but it might be hard to find. With the simulation method, instead, one can think of heuristics as being balanced against one another.

2. Heuristics are arguably under-specific in the sense of not stating every aspect of atom behaviour. The simulation approach says much more about it, in a sense adding far more heuristics. A very positive consequence of this was the noting of non proline cis peptides
3. What would sometimes be said in a heuristic, if said as part of the simulation, can be said in greater detail. For example, instead of having a “cutoff” distance from an amide plane, there may be a rigorous probability distribution

On the other hand, it is extremely slow, in the sense that atoms take quite a long time to relax to true energy minima. I believe the heuristic method is probably more effective for the time being, and may be effective all the way up until the point at which the model-refinement process becomes completely automated.

4.4 Manual model building and refinement

Automatic programs for model refinement are quite successful, in the sense that they create models that fit the data well and are physically plausible. In principle it is possible for the entire model-refinement process to be automated, and that therefore the user need not actually be involved in the decision-making process regarding whether the atoms should be.

In practice though, it is widely agreed that in the building of any model, the model and data should at least be briefly compared with one another at every non-hydrogen atom, and this is generally done in Coot.

Almost all biomolecule datasets require some manual rebuilding, and it is likely to involve a large number of variables and decisions. In this thesis I am interested in whether virtual reality can improve the process of model refinement. I am therefore only interested in tasks that involve 3D visualization and *manual* manipulation of objects in 3D. Model building and refinement is done with many pieces of software, most of which are *not* manual, but the manual part of it takes up a larger amount of time for users than the rest. The part of it that Coot takes care of is where the large majority of manual model changes occur.

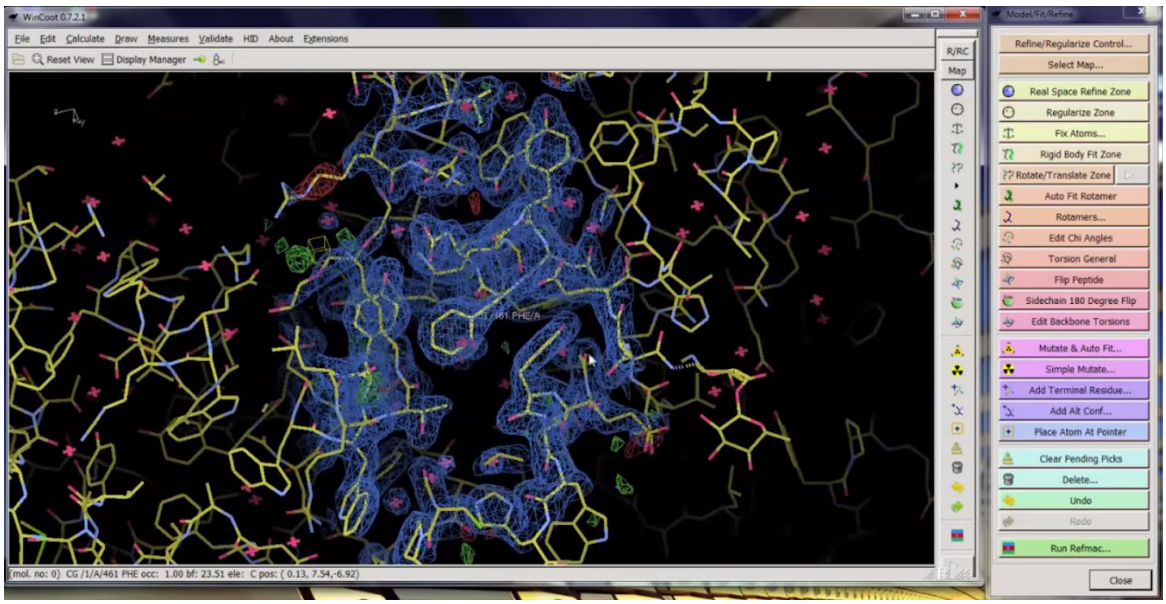


Figure 21: Coot, the ubiquitous software for manual model refinement. It has an ordinary icons, windows and menus mouse-and-keyboard interface, together with a 3D visualization window.

In Coot, the user visualizes the data and the current best guess of atom locations superimposed on each other in 3D. This allows the user to easily see any problems that may exist with the model, and move the atoms if there are any - Coot provides many tools for doing this as efficiently as possible. This process, which is also the purpose of CootVR, shall hereafter be referred to as “manual model building”.

4.4.1 Subjectivity and human error

Manual model refinement is a task with a particular role in the structural biology community. In comparison with automatic algorithms, it is a place where there is arguably more room for user error, i.e. a lab may make errors that lead them to believe incorrect things until they are corrected. For example, from misinterpreting noise in a dataset and misinterpreting sloppy about model-fit metrics, one may get the impression that a hormone is binding to a protein in a place that it is not actually binding. If academic reviewers do not look at the model-map-fit situation in Coot (which is time consuming), the error may never be corrected, and instead make it into publication where its truth becomes dogma in the field. This can waste the time of dozens or hundreds of people; for it to be corrected, the burden of evidence is put on the people trying to contradict it.

Bad situations such as this are more likely to happen it manual tools are more complicated. Of course, it is important to have a generally high standard for published research. But some responsibility for bad decisions lies with the software developers that enable them. And the ideal situation is to have no need for manual tools at all - if all drugs are fitted automatically, their model-fit metrics can all easily be held to the same standard. So automation is not just about time-saving, but allows for standardization and clarity. To give an example, Refmac is a program

that does model refinement, which is generally alternated with model building, and is completely automatic, and this means that if two studies' model-building process have used Refmac version 2.2, one can be assured that they are as correct as one another. Even if Refmac version 2.2 turns out to have a bug that caused it to make erroneous conclusions, at least the error can be documented and anyone wanting to use either of the models knows that they should first check that the bug has not affected the inference they are going to make.

However, doing these tasks manually, for the time being, does not appear to be avoidable if a high-quality model is desired. The problem is too complex to be *fully* solved from first principles (machine learning would be a good candidate for automating model-building, but there is currently not enough training data for it). If one day model building becomes automated, then it is our position that it would be a good thing.

5 Proposed advantages of VR

Across the virtual reality industry, there are a number of arguments given for the use of VR as opposed to a mouse and keyboard interface, which I will review here. Not all arguments apply to all software - many VR programs exist for the purpose of entertainment, a goal that, as described above, I regard as being orthogonal, if not opposed, to the practical concerns of scientific software. A more relevant comparison for structural biology software is another relatively common application of VR, which is CAD and artistic modelling. For example, the VR-based “Oculus medium” has been claimed to be 80% faster[60] than mouse-and-keyboard based modelling programs because of the gains of the arguments I will now outline.

5.1 Faster to “read” 3D situation

The world that human vision evolved to deal with is 3D. But modern software, when run on a conventional computer screen, will always be limited in what it can offer to our sense of vision.

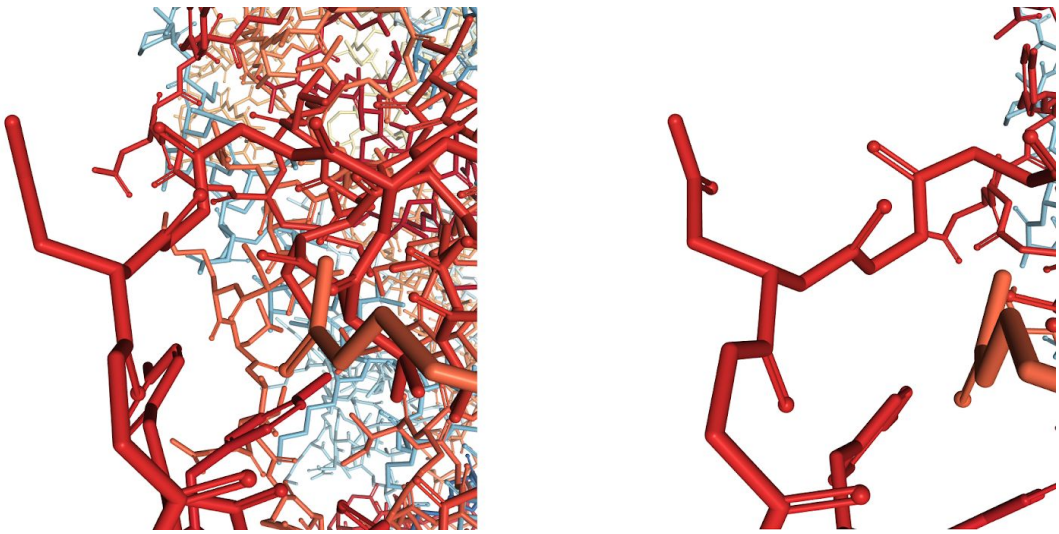


Figure 22: a representation of a set of atoms in 3D in NGL.

Figure 22 shows a simple example of non-ideal visualization. On the left, it looks as though a single atom is being looked at. But this is not the case; by moving the camera (the view on the right) we see that it is actually a pair of atoms and another atom was obscured. Such confusions arise multiple times per minute when using 3D software, needing camera movement for clarification. Ideally the confusion would not come about in the first place.

The above “problem” may seem very trivial, since it is resolved in a matter of seconds. However, it happens almost constantly. During frequent 3D modelling tasks, Coot users must engage in a very particular procedure: they must click on some menu items that do something to the model; move the camera over to the model and “shake” it to get the 3D impression of the state; move

the mouse back to the many they were clicking on and do more; move the mouse back to the molecule and “shake” again; and so on. It is not at all uncommon to shake the view several times per minute.

The momentary “confusion” and time consumption simply does not have to happen with VR. The reason for this is that the user can continually move their head (stereoscopic monitors mitigate part of it, but not all). The user can potentially move their head *while* they do something else with their hands. Even aside from these specific time-saving advantages, and other things being equal, one could reasonably expect that at least some tasks in Coot would be sped up by this aspect of VR. Relatedly, when users can more easily keep a good understanding of “where they are in the molecule” with the improved 3D awareness.

5.2 Improved three-dimensional articulation

When a professional wants to give spatial input into a computer program, they will use a mouse (possibly a touchscreen or joystick will be used, which amount to the same situation for many intents and purposes). This input is fundamentally two-dimensional - the mouse can move up, down, left, and right, and it works very quickly and intuitively, mapping very well to the fact that our vision is also fundamentally two-dimensional. However, if the user is moving an object they are limited by this. If they are orienting an object, the limitation is even greater: in the real world, untethered objects have three degrees of rotational freedom. In order to perform truly 3D tasks with a mouse, one must therefore bring in extra complexity, such as sliders or camera controls or extra keyboard commands.

Virtual reality offers an enormous improvement on this: the user can move their hand to an arbitrary place in 3D space, and reorient to any orientation they wish (technologies taking this even further are discussed below). This allows for software tools in which the user inputs a sophisticated three-dimensional gesture or shape such as a curved ribbon or even surface. This is still possible with a mouse, but it may be an order of magnitude slower. Again it seems reasonable to suspect that such capabilities ought to be useful, in some broad sense in structural biology, where I am concerned with 3D positions of large numbers of connected atoms, which may even be moving.

At the same time it should be noted that a hand controller can be worse than a mouse in some ways. Consider the following advantages of the computer mouse:

1. If a user moves their hand a large amount, the input will be nonlinearly scaled up such that the mouse moves a large amount across the screen
2. If a user moves their hand a small amount, the input will be nonlinearly scaled down so that the movement will be very small, possibly only a pixel
3. Because of the friction of the mouse’s plastic against the surface it is on, any jitters that the human hand has will not be seen in the cursor position. The friction also makes it easier for the user to make a small motion.

The fundamental problem is that in VR, hand position is mapped one-to-one - wherever it is in the real world, it will be there in the virtual world. Nonlinear scaling is not really possible, because if any kind of “interpretation” of hand input was made, the user’s virtual hand would quickly end up in a different perceived location from their actual hand, creating discomfort and confusion and therefore inefficiency.

Jitters of the user’s hand are also represented in VR, because the user suspends their hand in mid-air, meaning there can be no such “resistance” as given by mouse-desk friction. Haptic feedback provided by hardware such as the Novint Falcon, discussed below, would remedy this while offering most of the benefits of VR, but has other disadvantages.

5.3 Screen space

Virtual reality, offers more *screen space* than a desktop or laptop screen, which I mean in the same way that a laptop computer offers more screen space than a smart phone. Screen space significantly affects user interface design; the limited size first computer screen used for a modern operating system, that of the Xerox Alto(figure 23), required the invention of windows and menus to deal with that limitation[61].

Figure 23: the Xerox Alto, the first mass-market personal computer, had an implementation of drop-down menus and multiple windows.

In theory this offers a great deal of utility[62]. Coot, like other pieces of very specialized software, offers an extremely large number of possible tools and options. No user will use all of them, but

typical users will use a large number, and so must be given the option of using essentially all of them.

In a graphical user interface for a complex program, if all options are listed, they may take up essentially the entire computer screen or more. Some options also involve graphical elements such as bar charts or ramachandran plots, which take up even more space than words.

The way that Coot solves this problem is the way that it has been solved on computer screens since the 1970s - there are “drop down menus” containing multiple options. The user clicks a word, which may use a tool, or open a new window, or causes a menu of other words they can click, and so on. Usually the number of “layers” of this is no more than 3.

The “drop down menu” approach is time-consuming though, because the user must spend time in cycles of “look for word”->”move mouse to word and click”[63]. With the greater amount of space afforded by VR, it can essentially be the case that “all drop downs are always down”.

Our attempt at an implementation that takes advantage of this is discussed later in the chapter on the “panel”; I find that in practice this argument should be considered provisional, since the field of view and resolution of modern headsets is far from perfect; this is discussed further below.

5.4 Minor advantages

There are a few other advantages, each of which are interesting to consider, but that I would not describe as a major motivation, either because they do not affect many people, or because it is somewhat dubious whether they affect anything positive at all. Nevertheless they are not unreasonable suggestions:

1. VR may make it easier to train people to use crystallographic software, because there is no view control other than what human beings do naturally. Additionally, with more annotations on the model, it may become possible for complex stereochemistry concepts to be conveyed within the interface. This may make it so that undergraduate students could be trusted to refine models, which is not currently the case.
2. VR can be used to explain model-fit situations to people who are unfamiliar with Coot controls or even structural biology as a whole. This advantage was pointed out by two users working within the pharmaceuticals industry. They stated that it was common for them to see decision-making about how promising a drug was come down to the assessment by people who were not familiar with model refinement. In this situation, it may be advantageous to have them look at the molecule in virtual reality, with the electron density map there to give them idea of confidence levels concerning the proposed atom positions. This could make their decisions more informed.
3. Confidentiality is a concern for some Coot users, and VR may in theory be able to help with this. The user in question worked on unpublished datasets which were extremely

sensitive (i.e. the dataset showed a potentially very profitable drug whose discovery was not in the public domain). He worked in an open-plan office with multiple people in the industry, but who should not be allowed to see the drugs - but every time they walked past his desk, there was a chance that they would, and "leak" their composition to another company. A virtual reality headset, he said, would make it easier for him to win a contract, as he could offer this increased level of security (a headset is cheaper than moving to a private office).

4. Making a variety of gestures with one's hands may be more healthy than the current mouse-and-keyboard way of interacting with a computer[63,64]. A great many people get repetitive strain injuries from mouse use, including many structural biologists, especially because of the "Coot shake" described above. It may also be more enjoyable to work more directly with one's hands.
5. Users may make more informed decisions, because the substantial processing power that their brain puts into muscle movement will be utilized in their consideration of atom placement, when usually it is only visual processing[39].

6 Hardware

“Virtual reality hardware” is a phrase that has meant different things at different times[65]. When I say virtual reality in this text, I mean specifically *current* virtual reality hardware, which is one iteration of a hardware idea that goes back to at least 1968[66] - and is expected to be improved on over the coming decades.

Our software was developed for a specific hardware platform accessible to consumers during our working period, which I will describe here. I will also discuss other hardware I have investigated.

6.1 Main platform: “6-dof tracked headset and hand controllers”

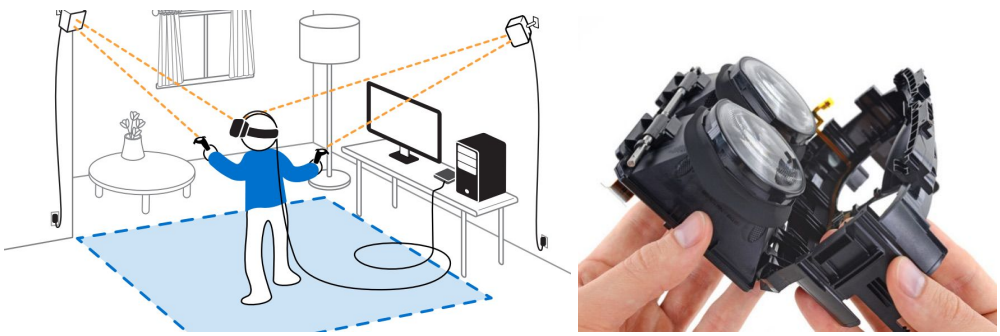


Figure 24: The major components of how the HTC Vive and Oculus Rift work

The products that I used were the “Oculus Rift CV1” and the “HTC Vive Pre”, which are identical in most ways. Their configuration is depicted in figure 24. They involve a single user donning a headset and holding two plastic devices with buttons on them. The headset is a plastic housing containing a pair of lenses and a screen, and is connected, by a long wire, to a computer. Also connected to the computer are two infared cameras that track the user as they are in virtual reality.

With this hardware and some software, it is possible to give the user the visual impression that they are in a virtual environment, and that they can see their hands in it. VR software consists of the following happening repeatedly in a fast loop:

1. The current positions and orientations of the controllers are acquired by the infared cameras, and their position is the real world is transformed to give a position where they *would* be in a “virtual world”.
2. The “virtual world” is “simulated” a single timestep. This may involve, for example, the hand hitting a 3D object in the virtual world and the object bouncing off it.

3. The position and orientation of the headset is checked using the same method as the controllers. From this a viewpoint in the virtual world can be determined, i.e. a head in the virtual world with a certain position and orientation.
4. A “frame” is calculated that shows what the virtual environment looks like from that viewpoint (in fact, it is calculated twice, one each for a pair of “cameras” located in the virtual world where the user’s eyes are expected to be)
5. This frame is sent to the headset, where the user will see it

Provided that the above happens frequently enough (essentially 90 times per second, discussed below), it will be the case that at every instant the user perceives, they are seeing *what they would see if they were in the virtual environment.* This causes their subconscious mind to make the inference that *they must be in* the virtual environment. They therefore feel that they can navigate it and move their hands around in the way they “naturally” do in the real world.

This does not work perfectly, however, and from the description it is possible to get the impression that the hardware is capable of simulating reality completely. This is far from the case, and so it is worthwhile to be aware of all the limitations, in addition to the drawbacks of the platform (considering especially the perspective of structural biologists).

6.2 Limitations of current platform

It is worth stating specific things the target platform *cannot* do. All of the problems below are things that will be solved in a few years, see “other hardware platforms” below, but it is important to be realistic about the proposition of actually using CootVR, which means using the current hardware.

6.2.1 Operating system

A simple but enormous problem with current VR is that it is Windows-only; this puts off many structural biologists from using it. It can reasonably be expected that this problem will be solved, but it is very difficult to put a timeline on. Coot’s usership is, by internal estimates, split equally between Linux, Windows and Mac. One option is to set up a computer just to use CootVR, which has been done by one lab, but this is very costly.

6.2.2 Resolution

The resolution of the screen inside the headset is limited, currently around 1080x1200 pixels per eye. The lenses in front of the screen make it make it worse; the user can easily see pixels. They also have a limited “field of view” of around 90 degrees vertically; this gives the feeling, when inside the virtual world, of having one’s vision blocked by ski goggles, in addition to seeing the world through a sieve or something else that pixellates it.

Because the screen is so close to the eyes, the resolution of the headset is actually worse than the resolution of an ordinary computer screen with the same number of pixels. As a result, when

words or pictures are displayed in the virtual world, they have to take up more of the user's field of vision than they would on an ordinary screen, otherwise they will appear blurred. This means that arguments in favour of VR that are variants of the "screen space" argument outlined above are currently not very applicable. Even though the user has more physical space that their "display" is occupying, the amount of information conveyed to their eyes is the same, because the resolution of the headset is comparable to the resolution of a monitor.

6.2.3 Material costs

A VR headset costs around £400. It has a certain minimum specification for the computer it is to be used with. The headset and hand controllers are quite a large objects and take up a moderate amount of room on a desk or in a drawer, as do the sensors if they are being used, so VR is not an option for someone with insufficient desk space.

6.2.4 Eye and muscle strain

VR causes strain on the body that leads to discomfort over time. This is not a huge problem if the hardware is being used for entertainment, which typically lasts no more than 2 hours. But the strain will be felt considerably if the user is in VR for 7 hours or more - and 7 hours can be a fairly typical working day for a user to be using normal Coot. The strain occurs in a few ways:

1. The eyes are having light shone directly at them, and are inside a headset, which can dehydrate them
2. The headset can weigh approximately half a kilogram, more than 10% the weight of the average person's head; this places strain on the neck
3. The controllers also weigh several hundred grams, and most important movements that involve them are more vigorous than movements needed with a mouse and keyboard, where the user's wrists can sit on a surface, and the rest of the arm does not have to have its muscles taught.

To help this, I have been careful to make sure that the user does not have to keep too many buttons held down. Additionally, I believe that headsets will become more comfortable in future; certainly eye-strain will be reduced by the creation of focus-sensetive displays, discussed below, and reducing weight is a major priority for hardware companies. However, even minimal strain is a very serious issue.

6.2.5 Time investment and "donning"

There is a time cost to using VR on top of a conventional screen setup. It is our belief that eventually transparent VR headsets ("AR headsets") will replace computer screens, and so there will be no time lost in using VR software as opposed to non-VR software. This is discussed below, but it is not remotely a serious proposition for consumers, and until it is the time cost of using both a monitor and a headset is a serious factor (the main negative factor) of using VR.

Firstly, it takes a certain amount of time to set up the headset, i.e. plugging it in and making sure the software is downloaded, installed, and working - and kept updated. The hand controllers also need to be charged or have their batteries replaced occasionally. I estimate that in total, this time adds up to perhaps 2 days per year.

There is another, more serious time cost incurred though. Going from sitting at one's desk not in VR to being fully inside VR and able to work, which I call "donning" the apparatus, takes a certain amount of time. The user must pick up the headset (potentially out of a drawer), put the strap behind their head, and adjust the front of the headset until their eyes are in exactly the correct place. They also have to pick up both of the hand controllers and make sure their fingers are in the correct positions; this can require them to briefly pull up the headset to find the controllers. In total I have found that this takes on average about 25 seconds per don, which compares unfavorably with, for example, putting on headphones, a common activity people do at their desks, which can take only 5 seconds.

Even the above assumes that the user already has the software set up and there are no hitches or extra buttons to press to enter VR, which is a generous assumption. They may find that the lenses need cleaning, and the process may be made much more complex if they wear glasses that they would like to have in the headset (although one can have prescription lenses fitted).

Donning and un-donning the headset is, for VR hardware currently on the market, a regular activity. Anything in the real world demanding visual attention requires un-donning and re-donning the headset, including drinking coffee and speaking with colleagues. Additionally, things that the user needs to use their monitor for, such as reading scientific papers and using non-VR software (eg answering emails), also requires the user to take off their headset.

New hardware will have an impact on the donning problem. Figure 25 shows two relatively simple updates that could take the 25 seconds down to 10 seconds.

Figure 25: VR hardware designs, not currently on the market, proposed to reduce time spent donning by allowing the controllers and headset to mostly stay in place even while outside the virtual world. On the other hand, having the hardware on one's body permanently is extra weight.

6.2.6 Nausea and vomiting

"VR sickness" is a sense of nausea, sometimes leading to vomiting, that is associated with some VR experiences. The reports on the very first piece of VR software for structural biology highlighted it as a major problem[6]. Since there is a non-VR alternative to our software (normal Coot), and it is intended, potentially, to be used for hours, it was important that I avoid this.

VR sickness is comparable with travel sickness, and has an interesting biological explanation. There exist fungi that are lethal if ingested and are therefore important to avoid, but look similar enough to safe fungi that they were often accidentally eaten by our ancestors. Some of these fungi had another effect, which would kick in prior to death, which is that they would cause a slight distortion of the information coming from our vestibular system (sense of balance grounded in the ear). However, our visual system would be unaffected. This meant that an organism could survive swallowing such a mushroom if it vomited it up quickly. Therefore, our bodies induce vomiting if any mismatch is detected between information coming from our vestibular system and our visual system. Humans appear to have varying sensitivity to this[67].

CootVR has caused VR sickness only in myself while I was testing features and experienced bugs. Even people who have had a history with VR sickness in other apps have not experienced it with CootVR. I have kept it this way by sticking to certain principles:

1. The frame rate should be kept above 90fps. This appears to be useful because it is approximately the frequency with which the visual system checks how our body is oriented.
2. I establish a "horizon" and "up direction", because of a gradient in the sky and a checkerboard pattern on the ground. It is worth noting that these are the only non-interactive or "cosmetic" objects that the user sees in the virtual world. This is in contrast to most VR apps, which try to make the environment more believable and interesting by filling it with objects.
3. I do not move the user without them triggering movement
4. I ruled out making use of some ideas on the grounds that they were vomit-inducing, see below.

6.2.7 The stereoscopy / vergence problem

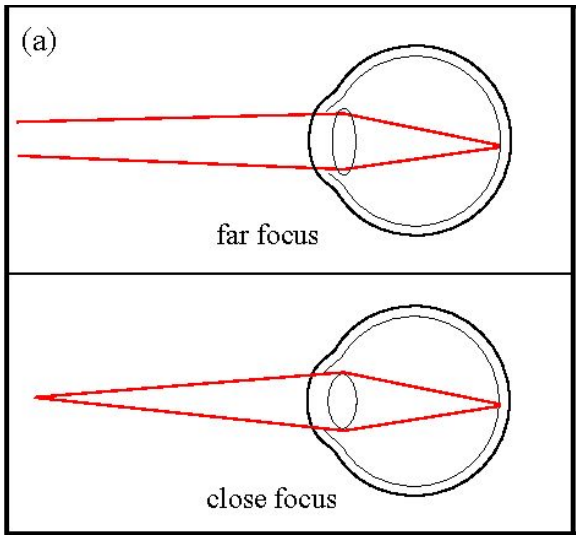


Figure 26: In order to focus on objects at different distances, our eyes change the shape of the lenses in them[68]

One of the major problems that has arisen in VR research, and remains unsolved, is called “vergence accomodation conflict”. Figure 26 depicts one subtlety of the way that eyes work; both eyes are trying to “focus” on an object from which light rays, represented by red lines, are coming from. Note that the “lens”, in the lower picture, is a little bigger than the one in the higher picture. This has happens because, in both pictures, the eyes want to make sure that the red lines coming to it get refracted to a specific place on the the back of the eye (which receives the light).

This system works mostly unconsciously (though one can choose to notice the effect by deliberately focusing one’s eyes on objects close or far away). Since it is not a conscious effort, our eyes “want” to quickly “choose” by themselves what to focus on. And one cue that they use to check for a lack of focus is how “crossed” they are, figure 27. Our brain subconsciously makes the following inference: if our eyes are crossed, then I must be looking at something that is close, and in fact from the angle of the crossing, I ought to be able to work out how close the object of focus is.

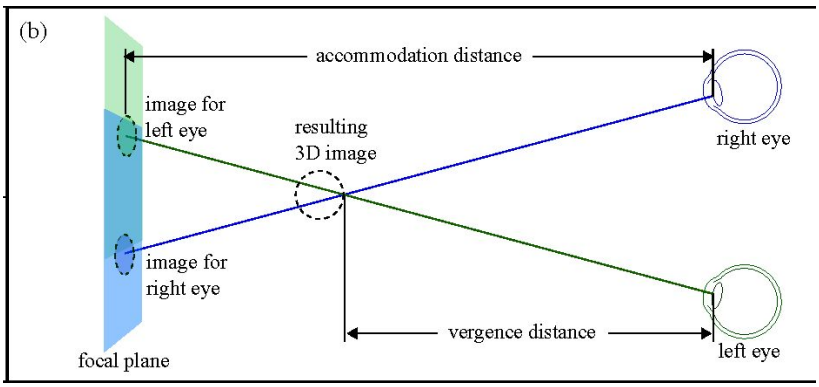


Figure 27: eyes cross in order to focus at an object at a certain distance

This unconscious system works usefully and perfectly *because it is a reliable and logical deduction* that if one's eyes are crossed at a certain angle, there is only one possible distance that the object they are focussed on can be at, from it can be deduced how much the lens should be compressed to focus the light from the object onto the retina. But counter-intuitively, it has the opposite of the intended effect if one is using a current-generation VR headset.

In the Oculus Rift and HTC Vive, the *reality* is that all light is coming from a screen inside the headset, which has one specific distance from one's eyes, and objects only *seem* to be further away or closer than that distance. If one is looking at an object more than a meter away, this is not a problem - our eyes are willing to focus on objects that distance away. However, if the object that our eyes want to focus on is close enough that our eyes become even slightly crossed, then the "vergence accomodation" problem comes up.

As usual our eyes make the deduction that the light must be coming from an object eg 30cm away, so they refocus. But it is not; the light is coming from the screen, which is (always) less than 30cm away. Thus, the light does not get focused precisely on the retina. And because the light is not focussed on the retina, everything appears blurred. Extremely exotic solutions to the problem are being attempted by tech companies and labs, for example having multiple transparent screens inside the headset, but nothing that is currently on the market, and it is unknown how long the problem will take to solve.

This is not necessarily a problem in many virtual reality programs, eg if things stay a large distance away from the user in the virtual world. However, bringing things close and working with them in detail is the whole point of CootVR. I therefore considered some solutions:

1. I tried making it so that objects brought close to the camera would disappear, discouraging the user from entering a situation where it became blurry. However, this was irritating, more so than the blurriness.
2. "Inter pupilary distance" controls were implemented. As described above, the image that the user sees is generated by placing two cameras in the virtual world side-by-side. Ordinarily they are spaced ~6cm apart, the average "interpupillary distance", but one can

put them closer, which removes blurring at the cost of removing stereoscopic information. However, this idea is self-defeating since stereoscopic information is the whole point of bringing something close to one's face.

3. I considered the extreme solution of manually crossing the user's eyes when they brought something close to their face. The idea is that the brain would receive stereoscopic information without the eyes having to cross. However, accidental eye-crossing is something that I experienced as a result of a bug at one point, and it was, unsurprisingly, very unpleasant.
4. Another extreme solution considered was a pair of spectacles to be put inside the headset for close-up work, but again this would be so inconvenient as to be not worth taking the headset off.

6.3 Other hardware platforms

The space of visual and interface hardware capable of improving the situation for model refinement is wide. I have experimented with and researched some of them, described below.

6.3.1 MobileVR

"MobileVR" is a type of VR that involves putting a consumer smartphone, equipped with a gyroscope and accelerometer, into a casing with lenses which is then attached to the face. This is not a description of one product, but of several dozen different products for which some standards exist.

Figure 28: MobileVR

MobileVR offers advantages in comparison with the Vive and Rift platform described above:

1. It can be used in conjunction with any operating system
2. It is much lighter on the head
3. It is (with some caveats) two orders of magnitude cheaper, £25 as opposed to £450+
4. It is does not have to be confined to a desk - in principle, one could use it to work on a structure on the bus.

Superficially, mobileVR is very similar to the hardware platform described above, and considering the points above, it may even seem superior. However, mobileVR has one enormous comparative disadvantage, which is that it does not have “position tracking”, only “orientation tracking”. This essentially means that the user can feel like they are in the virtual world, and can even “look around” in it, but if they *move* (not simply “tilt”) their head forwards or to left, nothing will happen, i.e. “the entire virtual world will move forward with them” as if they had not moved forward at all. Many mobileVR headsets come with a single hand controller, to which the same applies: one can have one’s hand in the virtual world, and can reorient it - but not exactly move it.

MobileVR should not be dismissed easily though, because there are ways to move around a space within it. For example it has been discovered that one can attach a mouse to a VR system, then move one’s head away from the mouse, but continue moving the mouse in the real world with it separated from its place in the virtual world[69]. Inspired by this, I decided to attempt to make CootVR work well with mobileVR platforms, reasoning that using a mouse and orientation-tracked controller together, I could emulate position tracking. Multiple setups were tried, including a solution in which I would “grab” an atom with our mouse and then move our head around that point by reorienting it. This may sound a strange way to move, but I reasoned that people ought to get used to it, since they are able to get used to other strange ways of manipulating space, for example rotating a handle to reel in a fishing rod.

However, on reflection, the kind of interaction that mobileVR offers simply does not add enough on top of mouse interaction to be worth using, especially not if the controls take extra time to learn to use. Additionally, mobileVR is a more complex platform to develop for, because phones do not have standardized dimensions, and because it is necessary to “broadcast” data to the (wireless) phone. For these reasons, I abandoned the mobileVR aspect of CootVR. One of the results of our experiments is available at hamishtodd1.github.io/Cardboard and the versions of CootVR that are compatible with mobileVR are available at all commits before this one https://github.com/hamishtodd1/CVR/commit/ff870c4173efe0f5bb67ad65e67f368cbdb3ba06

6.3.2 Mixed and augmented reality

Mixed and augmented reality (MR/AR) are when virtual objects are superimposed on the real world in some way. Comprehensively defining them in a more concrete way than that is difficult, so to ground our discussion I will refer to a specific MR/AR product that was on the market during our project called the Microsoft Hololens.

Figure 29: Microsoft hololens. It is similar to a VR headset, except that the display is transparent, meaning that virtual objects can appear “on top of” real world ones.

There are several problems with using HTC Vive or Oculus Rift at a desk job that could be solved immediately by a transparent display:

1. Users cannot see their keyboard or mouse which are useful for interaction
2. Users cannot switch to using other things at their desks such as pen and paper, a phone, stress balls, drinks containers etc.
3. Social interaction is almost completely ruled out; users cannot see facial expressions made by someone one is conversing with, and they are uncomfortable too as they cannot see your eyes.
4. Unpleasant feelings about VR can arise from the fact that the user is not seeing natural light; they are usually alone in the virtual world; and they are surrounded by many objects, but none of which are actually real

All of these are problems that one can solve by taking off the headset momentarily, but as described above, this can be an inconvenience - potentially enough of an inconvenience to cancel out the benefits of CootVR. On the other hand, the hololens has problems too. Its field of view is famously terrible, requiring the user to have their head directed straight at an object to see it at all. Crucially, it also lacks hand controls, which I argue are the main benefit of VR for model building.

The HTC Vive actually does have a camera built into the front, allowing the user to see out of the front. But this only works to a limit extent - the camera is not in the place where one’s eyes are, it has an uncomfortable delay, and it looks bizarre in the virtual world, being a strange screen floating in front of one’s face - consequently almost no developers have made use of it. The next generation of headsets will be better in this way, being able to construct a 3D representation of the world, so most of these problems will be solved, probably with the exception of the point about other people feeling uncomfortable talking to the user while they have the headset on.

6.3.3 Controller-free hand tracking

Figure 30: The “leap motion” is the state of the art of controller-free hand tracking

The leap motion, figure 30, can acquire the user’s hand and finger positions without need for a controllers. This has a number of advantages:

1. Since the user is giving more information as an input, they are able to articulate more sophisticated things. In our context, an interesting example might be to pinch a sugar pentose with two fingers and a phosphate group it is linked to in the same hand, and manipulate them semi-independently.
2. The controllers never need to be picked up or put down (see above, the “donning” problem)
3. It is easier to learn (this is not necessarily a hugely important argument though, see above)
4. The “precision grip” makes it easier to perform complex movements with one’s hand than the “power grip”[70,71]. With held controllers, all grips are power grips.

Other devices for the same thing exist, such as the Microsoft Kinect. However, all are quite high-latency, which is uncomfortable and mistake-inducing for prolonged usage. Michael Abrash, chief scientist at Oculus, has said that he expects rigid hand-attached controllers to be the main input device for virtual reality until around 2028[72]. He puts a similar estimate on how long it will take for haptic devices to become appealing to consumers. Hearing this and similar opinions made us decide not to pursue this technology further.

6.3.4 Haptic / “force” feedback

In the real world, almost all objects one encounters give “resistance” when touched, which can be felt with the nerves in one’s skin and muscles. From watching videos or hearing verbal description of modern VR, many people get the impression that this can be simulated - but it can’t be. This is the major place where current virtual reality feels unitinuitive and “fake”, and I have had to take it into account in much of our user interface design. Controllers can be made to vibrate, but this does not go very far to simulating “resistance”, so I have not made use of it.

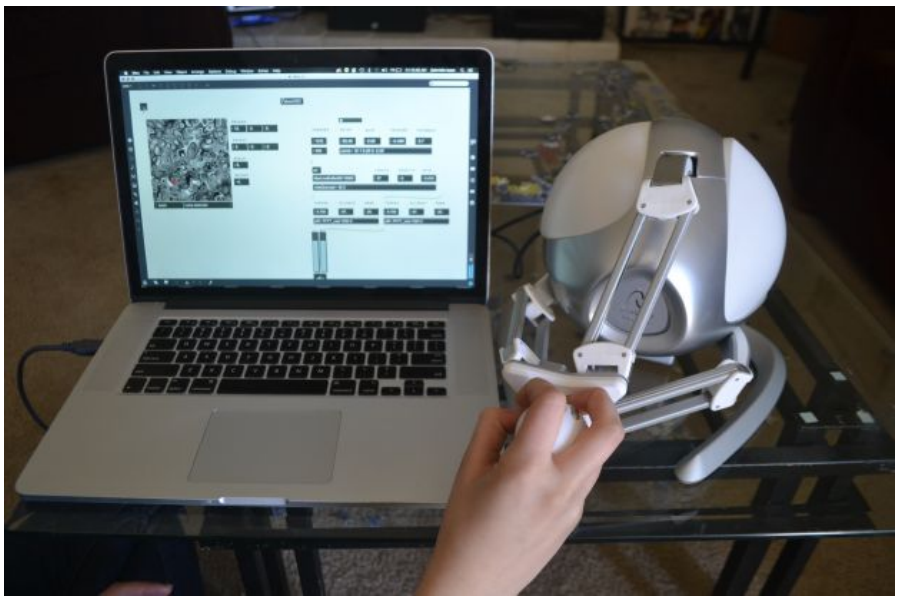


Figure 31: the Novint falcon being used for a scientific application[73]

Chemical constraints and electrostatic repulsion are certainly a “natural” match for haptic feedback, and in briefly when developing the CootVR features described below, I wished that VR had haptic feedback. Specifically I felt it would be good for communicating the position of the “nearest good fit” to a user-specified set of atom positions using force vectors; however, visualization does about as well.

There are hardware platforms that try to simulate “resistance”, which are referred to as “haptic” technologies. There are entire scientific fields dedicated to investigating haptic feedback, and multiple haptic devices do exist, using gloves, ultrasonic waves, air vortices, plastic controllers attached to large mechanical apparatus. However, the value proposition of currently-possible hardware has not yet made it worthwhile for many consumers. Nevertheless some model-refinement software has even been written for them, for example ISOLDE[34] which uses the Novint Falcon, figure 31.

However, these devices have historically suffered from a problem. With many of them, including the Novint Falcon, the user has to look at a representation of their hand on a screen, which can

create more problems for one's intuition than it solves. The only way to get around this is to have the haptic technology be an add-on for current VR technology, VR being essentially the only technology that can put the visual representation of one's hand in the place where it actually is in the real world. But a haptic device that is an add-on for VR technology add further monetary and time expenditure to the already-difficult proposition of a VR headset. In addition, most devices still have issues with "lag", making them unpleasant, or only of momentary interest.

6.3.5 Headset-free 3D enhancement

Specifically for the purpose of improving the impression that the audience receives of a 3D object displayed by a computer, several technologies have been developed, some of which have been utilized in model building.

"3D monitors" such as the Zalman ZM-M220W involve putting on polarized glasses and displaying two superimposed images, one for each eye, the same way that movie theatres simulate 3D. This has had some adoption in the structural biology community, mostly for using Coot. Anecdotally, I have found that for some users it is very important. However, for other users, its use is extremely marginal, and so in spite of owning a 3D monitor, they will not even bother to turn the 3D effect on. In developing our software, targeted at VR, this was useful to be aware of, as an example of a "fad" technology that was adopted only by people "excited" by claims that were not completely proven.

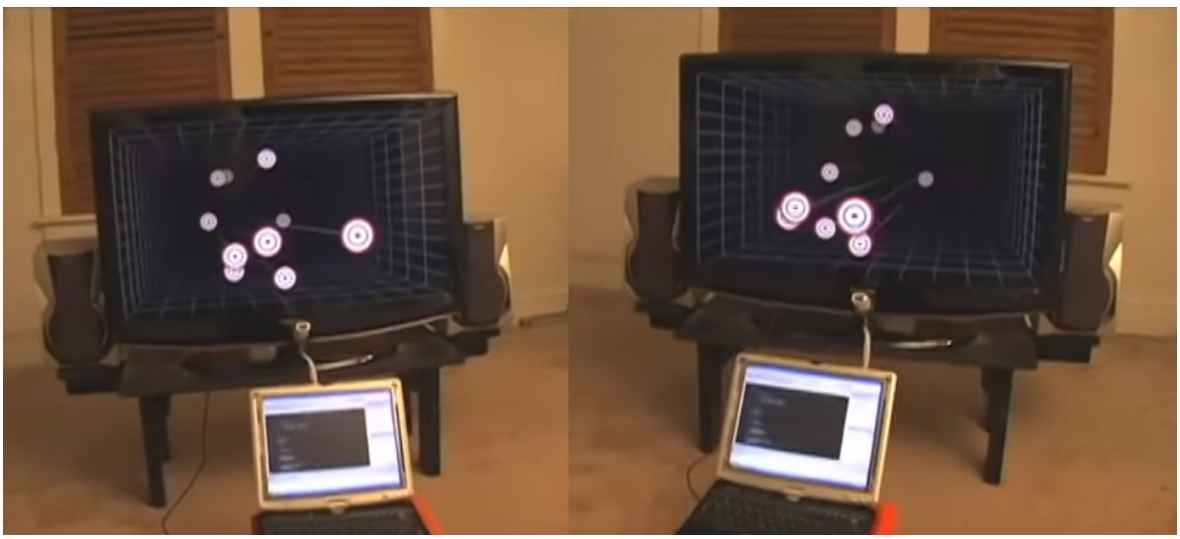


Figure 32: a basic "head tracking" display[74]. The targets are repositioning themselves on the screen according to the position of the camera. This creates the illusion of depth.

Another technology with the same goal, but with a very different means and mechanism, is using an ordinary monitor with head tracking, figure 32. For this to work there must be some means by which the monitor can know at least the left-right position of the user's head, which

can be achieved simply with a consumer webcam. With this setup, if the user keeps their head still whilst using the display, nothing will happen and they will see the 3D screen as normal. If they move their head however, the scene will be re-rendered with the virtual “camera” placed at a different position. Ideally, this has the effect of making the screen seem like a “window” through which a virtual world is being viewed.

A technology that elegantly combines both of these approaches is the “light field display”[75]. This very recent innovation is a screen with pixels that emit different colors in different directions; consequently, two different perspectives on the same screen at the same time will see two different pictures. The images being displayed can be such that they are a single 3D scene (eg a molecule and map) rendered from every possible possible viewing angle. This allows viewers to get information from head movement *and* stereoscopy, because our left and right eyes will be viewing the scene from their exact vantage point. Consequently, nothing head-mounted is required (not even 3D glasses), and multiple people can experience the enhancement.

Headset-free tracking is a promising technology and I fully expect lightfield displays to be used by the structural biology community. However, they are not currently available to consumers. Additionally, our main area of interest is the way that users use their hands, which ideally involves the user’s real-world hand appearing in the same place as the virtual-world hand. I believe that, once large consumer light field displays can replace monitors, for those not wanting to use a headset, a light field display with a hand controller attached to the back might be a good alternative to VR.

Part II

Supplementary material: “CootVr Basics” video on youtube:
https://www.youtube.com/watch?v=TdyYOWKDpGc

7 Hand tools

The most interesting benefit of the VR platform for model refinement, in our view, is that the user can interact with the software using fully articulated hand movements. Here I describe those tools that benefit the most from hand movement.

All of the tools are “big picture” movements of a large number of atoms. This is to be expected: smaller groups of atoms (a single sidechain for example) can have their conformation worked out automatically with chemical constraints; and larger groups are things that there might exist more uncertainty about, i.e. there may be more room for the user to be vague about where things should be - hand movements cannot be expected to be precise to within a single angular degree. It is acceptable in the course of model refinement to have a model be *temporarily* not-exactly-correct - it is a “price worth paying” to be able to quickly examine different possibilities that are “vaguely correct”. If these possibilities, which will mostly be wrong, are not at least examined, it is possible that a superior model-fit situation will not be considered; or that it will take longer to find, because a lot of time must first be spent ruling out incorrect possibilities (because they must be considered in great detail).

All of the tools described here are selected from the panel, see chapter 8. Different ways of selecting them were thoroughly considered. The state of holding a tool in one’s hand is arguably a “mode” similar to “insertion mode” in older text editors, and is something heavily warned against by some UI designers[76]. A possible alternative to “picking up tools” would be to have the model covered with small handles (which need only appear when the hand is close to them) which could be grabbed and moved in order to have the same effect as the tool. This would not work for all tools though, for example the painter tools.

7.1 Examining the model and map with different orientation, position, and scale

The simple act of examining the molecule and map with one’s hands, in order to see it at different positions and orientations, is an extremely obvious use of VR. It is quite beneficial, and has some subtleties to its benefits.

The way that it works is that, when the user “grabs” the model and map, they will become “stuck” to the grabbing hand. If the grabbing hand is moved, the model will move precisely such that the grabbing hand stays at the same position and orientation relative to the model.

Additionally, the user can grab the model with both hands and “scale” (with the right hand being the center of the scaling; this allows for easy control over one’s “focus”). This is somewhat analogous to a “zoom” on an ordinary screen (especially within Coot), but not exactly analogous because a zoom is often thought of as moving towards, or away from, what one is looking at.

Scaling the model in VR breaks this analogy: the user perceives the object in front of them to be staying in the same place, while inflating and deflating. Users can *also* move towards and away from the model in order to get a better idea of its larger or smaller features, and in practice this is an important thing that happens all the time, but its effect is much smaller than what one can do with zooming, and if the user gets too far away then their arms cannot reach the atoms - so a combination of the two is very natural.

The purpose of rotating and moving the molecule and map in Coot and CootVR is (obviously) to change what one is seeing - and I claim that the user can achieve this goal more quickly in CootVR. Even for an expert user of the Coot view controls, it takes longer to get to the view that one precisely wants than with CootVR. I would estimate that during Coot use, around 1-2% of the user's time is spent adjusting the view, although this goes up to 5 or 6% if I include the Coot shake that CootVR eliminates. On this front therefore, CootVR is a considerable speedup.

A user can also "multitask" to a great extent. Using other hand tools, it is very natural and fast to be doing something to the model with one hand, while the other hand is holding it and moving it such that the doing-hand is in the easiest place possible and the head has a good view of what they are doing. This is in contrast with Coot, where view-changes are something that must be done between tasks - potentially it is the case that what should be a single movement is broken up into many movements by Coot shakes.

One major architectural difference between the Coot and CootVR user interfaces is that Coot has a "residue of focus", which is the residue at the center of the screen. The user has two keyboard buttons that they can press to go to the "next" and "previous" residue on the chain. I have not implemented any such thing in CootVR, and have the strong intuition that it would be a bad idea. In Coot, the "chain" is thought of, to some extent, as being in a flat plane. In CootVR, it seems likely that the user will want to work on 3D neighbourhoods, as opposed to flat slice-like neighbourhoods. I believe this will be more useful for general changes to the positions of large groups of atoms (entire alpha helices for example), such as those needed for Cryo-EM work.

In this thesis I have focussed on "model refinement", although that is not the only thing that Coot is used for - it is also useful for "analysis", i.e. examining the model and figuring out what information it gives about the activity of the protein. CootVR can in principle be better for analysis than Coot, because it involves a great deal of view-adjustment, and examination, although no moving of atoms and therefore not much hand-usage apart from the view-adjustment.

7.2 Rigid mover

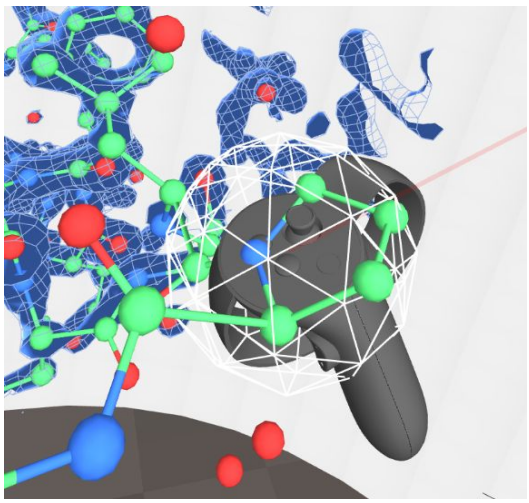


Figure 33: the user's right hand, holding a histidine residue.

"Rigid" motion is when an object moves as a "rigid body", i.e. as if it is completely frozen and not a single part of it is moving with respect to any other part. It is used sometimes in Coot for, for example, moving ligands. Coot also features an automatic rigid fitting tool, but it is not always the best option, especially when resolution is low or changes are expected to be made to the rigid atom positions being fitted. I have implemented rigid motion in CootVR using the hand controllers. This allows the user to perform a rigid motion with, in principle, any set of atoms (figure 33).

Rigid motion is an extremely simple kind of movement and completely insensitive to context. When rigidly moving atoms, there will be many "problems", for example steric clashes, or the breaking up of a beta sheet that changes the allowable ramachandran for an amino acid. But these can be dealt with separately; in general the point of rigid movement is to put atoms in an approximately on-the-whole better position, see figure 34.

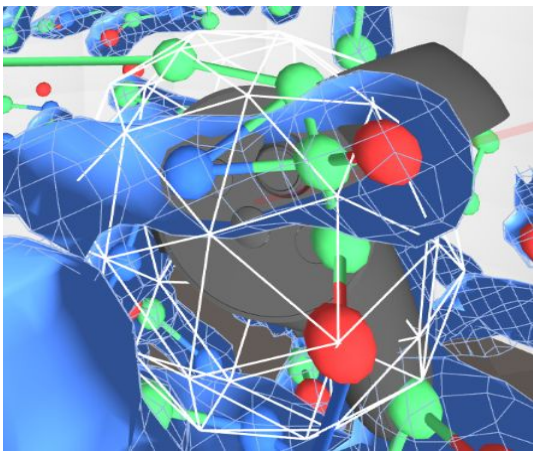
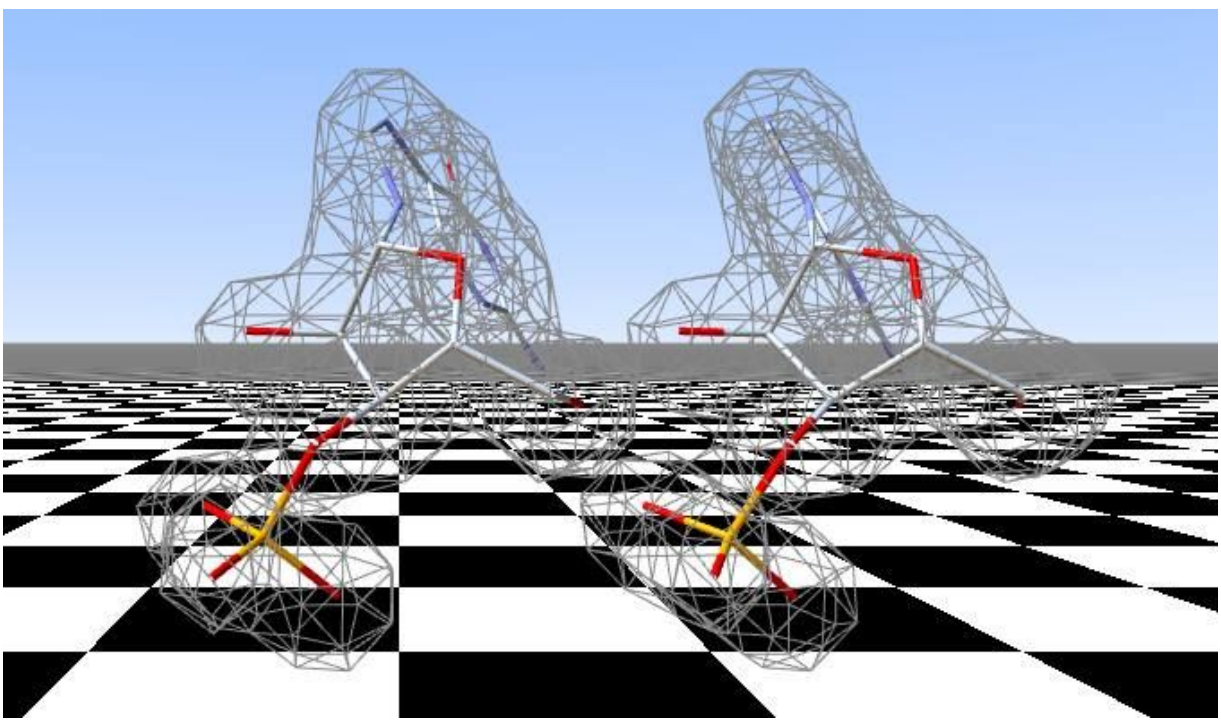


Figure 34: the rigid mover holding parts of a model into place in a map. The cage-like appearance makes it relatively easy to keep track of what is happenning.

One example is, again, ligands, where it is extremely common for the user to want to pick up a specific set of connected atoms within a region and rotate and translate them. This was actually the first interaction that was programmed and performed using CootVR, and is often the first thing that users pinpoint as something they would like to use CootVR for.



To give a more interesting example, one may be considering the fit of a homology structure into a map, and have the vague feeling that an alpha helix ought to be moved. Doing this “properly”, in the sense of carefully making sure every amino acid fits, may be worthwhile, but it is not yet

clear that that would be worth the time investment. All that is desired, for the time being, is to see whether there is room for an alpha helix in a given place. The rigid mover allows this hypothesis to be tested in moments, where previously it might take at least 5 minutes.

Two different area-selection methods were implemented, sphere-selection and chain-selection. With sphere selection, when the user presses the grab button, they will be holding a set of atoms that are inside a sphere. The sphere is cage-like so that it is obvious which ones are contained within it. The chain-selecting tool works differently: the user must put both of their hands somewhere on the chain, at the start and end of the part of it that they would like to move. All the amino acids “between” their hands’ position on the chain become selected, and they can move the chain around as desired.

In principle, the rigid mover has a use beyond model fitting, which is communication. Consider two structural biologists talking about a piece of chemical activity involving a few dozen atoms. One of them has a hypothesis about how the chemical activity takes place, such as “this amino acid moves, then this one vibrates, and then if they work at the same time then it allows the ligand to bend this torsion angle”. However, they are having difficulty communicating it to the other biologist. One aid to them might be making dozens of drawings with pencil and paper. However, simply by using the rigid mover, they could illustrate the same series of actions in a smaller amount of time, with a very intuitive tool. In this situation, having a molecular dynamics simulation changing the atoms might be a positive thing, as other studies have suggested, but that is not necessarily the case.

With the spherical rigid mover tool, it is important for the user to be able to choose the number of atoms that they are grabbing. Our initial idea in this direction was to have a sphere that is moved by both hands, one holding one side, the other holds the other, and the user could scale the sphere by moving their hands towards or away from one another. While intuitive, it is cumbersome to have to involve both hands.

In the end I opted for a simpler approach with essentially the same implication: that the rigid mover sphere would stay the same size, and if the user wants to increase or decrease the number of atoms fitting inside it, they would have to scale the model. This was partly based on a small but quite illuminating experiment that I performed early on in the project: I had a map and a molecule that could be lined up perfectly, and I was attempting to rotate and position them such that they did line up. I tried them at two different scale levels, one very large (2 meters across) and one small (10 centimeters). Lining them up at the large scale level (i.e. I was manipulating enormous objects) was incredibly difficult; I attempted it multiple times over the course of 15 minutes and succeeded only once. But at the small scale level it was extremely easy, and could be done in seconds.

When the atoms are grabbed, I allow the user to move them in a completely unconstrained way. I could have, for example, made all bonds into springs and when the user pulls on them it will pull on other atoms as necessary, or made it so that steric clashes are impossible. The

argument for this is that it could help keep physical realism; it might even make it so that the user could move the atoms around to random places to “see where they can fit”. But imposing constraints could be a hindrance too, because it might create the need to move a part of the chain out of the way before moving another part to where they want it.

If the project were to be continued, one of the most worthwhile things to have would be a mover similar to the rigid one which preserves, at least, bond lengths, which would give the feeling of moving a protein chain around like a heavy rope.

7.3 Protein painter

Since a protein is technically a chain contorted in a particular way in 3D space, a very obvious way to create (and indeed to speak about them) is for a person to move their hand through the air, tracing out its shape (“painting” it). This has a clear application within Coot as well: it is relatively common to want to create a chain of a specified length with some a specified geometry. One of the first serious tasks identified for CootVR was using such a tool: it is quite common to have a structure such that there is a bundle of alpha helices that are known to be connected in some way, but it is not known how. In this situation it may be necessary to try out many possible different ways to connect up the helices.

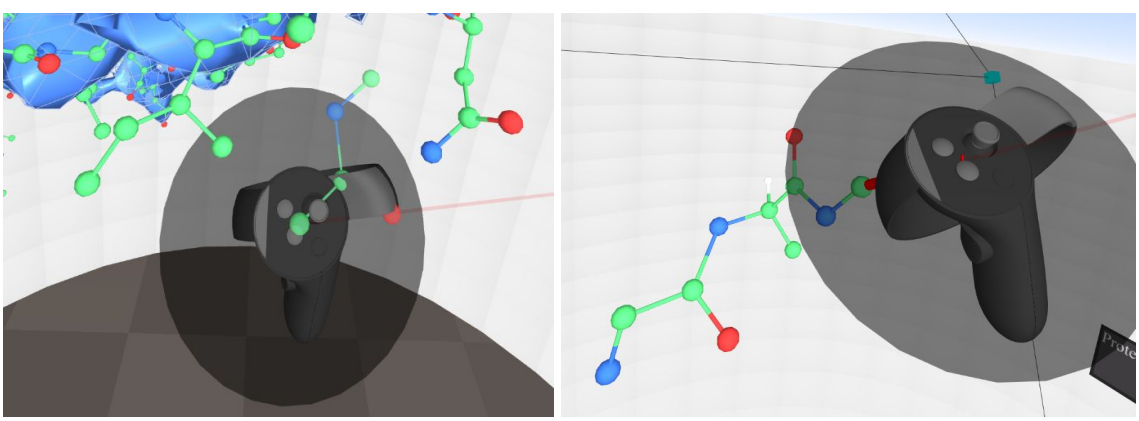


Figure 35: The protein painter about to lay down a first amide (left) and in the process of being used (right).

The geometry of protein backbones is well characterized. There is a reliable abstraction that is used to talk about it known in the literature as “amide planes” or “torsion angles”; our tool makes use of this formalism; see figure 36 for an illustration of it. In the formalism, the nitrogen-carbon alpha bond becomes a natural object to “pivot” around.

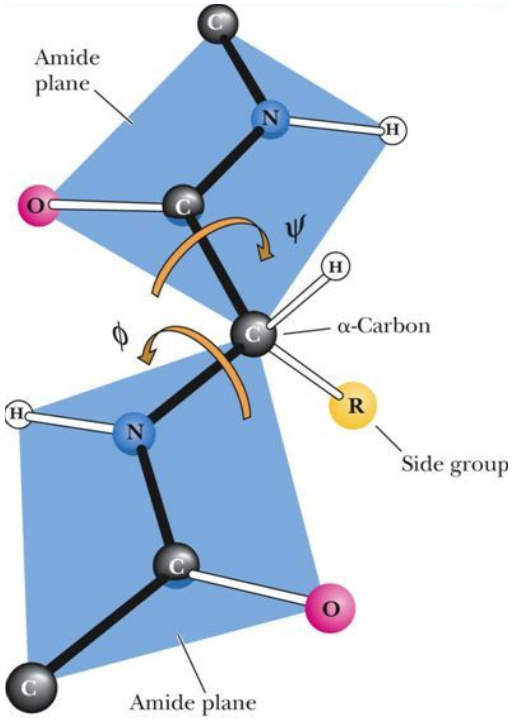


Figure 36: all angles in this image are completely determined by interatomic forces, with the exception of Φ and Ψ, which, if known for every amino acid, determine the positions of all the atoms in the backbone. The blue rectangles are “amide planes”, an abstraction that allows a protein to be broken up into completely-constrained pieces. The unmarked angle between the two bonds with torsion marks is relevant too, the so-called τ angle, which is known to change from the usual 109° during protein folding. This third degree of freedom is somewhat relevant in the discussion below.

The painter tool is the most original and sophisticated tool within CootVR, and went through an extremely large number of iterations. It is very easy to use and specific to VR, and has been the main tool in our case study below. Its design has been non-trivial because of the question of exactly how to allow the user to be “expressive” with it, i.e. allow them to create the geometry they want to create as quickly as possible, but also to enforce the chemical constraints of the phi-psi formalism.

The way that the painter tool has ended up working is as follows:

1. There is a button the user may press to create a new amino acid and immediately assume hand control of it. If there is already an amino acid under hand control when the button is pressed, it will become frozen it in place, and the new one will be attached to it with appropriate bond angles.
2. Another button reverses the above, essentially an “undo”, deleting the current amino acid and resuming control of the previous one (if it exists)

3. If the user moves their hand around, the phi and psi angles of the currently selected amio acid will change such that its alpha carbon will be as close as possible to the user's hand position in 3D space.
4. If there are two possible sets of bond angles that will achieve equally-close proximity to the hand, there is another button which, if pressed, will switch between these two sets.
5. While doing this, the user can move their head as they please, and move the molecule too.

The above strikes a balance of being extremely fast to use (see the experimental timing section below) and staying within chemical constraints to a large extent. Our initial ideas were very different though:

1. Our first version was to be based on the characteristic "bottle opening" motion that people do with their hands for twisting open bottles. This is not really possible to replicate without haptic feedback though. It is also not clear how it allows a user to change two torsion angles at once, which I wanted to enable.
2. Our second idea was based on a "rudder", see figure 37. This was implemented, but was very difficult to use.
3. I changed the visualization of the rudder by making a copy of the held bond appear by the controlling hand; this was also difficult to use. After this I decided that requiring the user to hold the object with both hands created too much complexity, although it remains an option.

Figure 37: The "rudder" design of an early version - the lower hand is moving the bond it is touching as a rigid body, while the upper hand, if turned left or right, can change the torsion angle of the "Y" with the white plane remaining rigid.

After this point the idea I converged on was variations on the user controlling the phi and psi angles completely using the orientation of their hand:

1. I extracted a "specific-axis orientation differential" from every hand movement as a quaternion and applied this to the amino acid, while enforcing that its tau angle remain 109°. This was somewhat intuitive but never felt like it was happening at the correct speed, even when I applied a scalar multiple to the angle.

2. Instead I treated the user's hand as a plane, extracting the yaw and pitch and applying them to the amino acid (again making sure to keep tau angle 109°).

The hope implicit in the above is that humans have a deep intuitive understanding of the orientation of their hand (or rather the topological space known as "RP3" which is homeomorphic to it), and therefore they can easily understand any simple mapping from RP3 to some set of possibilities they are want to explore. With the benefit of hindsight I can say that this was an overestimate of human intuition.

After the failure of this approach I decided to try to simplify, which resulted in the design described above: that the carbon alpha of the currently-moving amino acid will just "try" to go as close to the user's hand as possible, and the phi and psi values will change accordingly. This means that the orientation of the hand it is trying to follow is irrelevant. It seems to us counter-intuitive that this is the best answer; our intuition in all the above was that because backbone geometry is not "about" *positions* as such, but rather about *angles*, and that therefore the hand orientation should be the way to control it. Nonetheless there can be a high level of confidence that our method is better than any orientation-based one.

The above concerns amino acid conformation, but other aspects of this tool went through major revisions too. For some time I aimed to avoid using buttons and have the tool work completely "gesturally": "retracting" amino acids would be done by going backwards; creating a new one would be done by simply moving one's hand further away from the current carbon alpha; and peptide-flipping would be accomplished by flicking one's wrist. I believed these would be better because they were faster. However, doing precise tasks with them was hard to get used to because some thought was needed as to exactly how one's gestures would be interpreted. They are all binary (rather than analogue) tasks and so are well suited to a button.

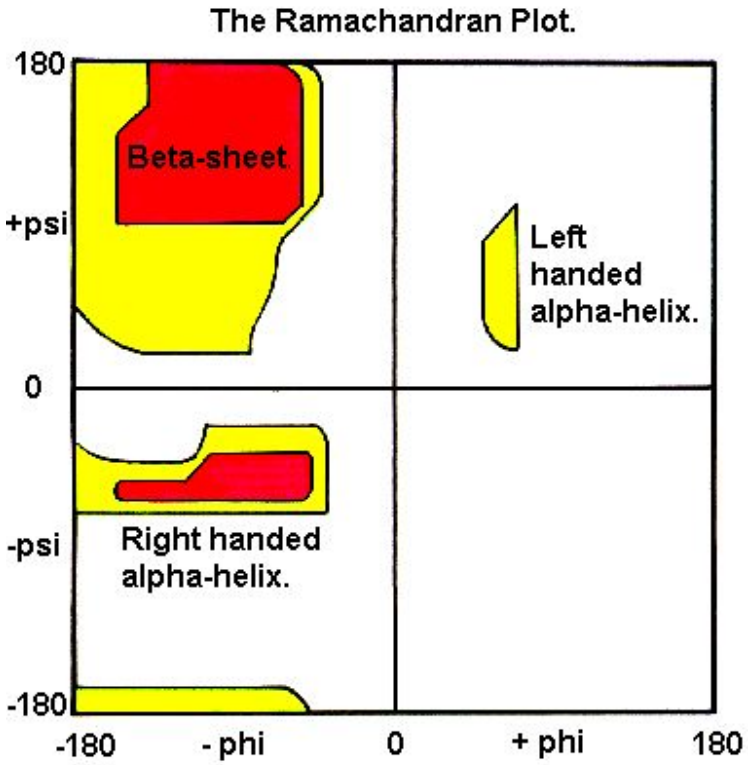


Figure 38: A basic ramachandran plot of phi and psi, from [77]. The highlighted regions are where amino acids conformations are generally found in experimentally determined structures - although theoretical methods like hard-sphere approximations may be used to conjecture plausible conformations too.

The tool I have created could be criticized for being insufficiently automatic. I did briefly consider a much simpler idea than all the above: having the user set a simple curve through space, and then filling it in automatically with protein chain, i.e. the user would need to have no direct experience of the chemical constraints. This would not have been especially difficult to implement (a Coot session would need to be connected to CootVR). I could have had mild automation, for example having peptide flipping be automatic rather than triggered by a button. But this would go against the idea of “experimentation” that I described above, with peptide flipping happening at, for the user, rather unpredictable times, creating a “fish slipping out of one’s grasp” feeling. I did consider visualizing, in place, the “hard spheres” used to make the original Ramachandran plots[77,78], or the Ramachandran plots themselves; I decided that this would not be very useful.

It should be noted that the protein painter, as implemented, makes chemically quite unrealistic chains and should not under any circumstances have models created purely using it be published; in addition to the above there is some unrealistic “allowance” in the tau angle that makes interaction using it easier but is undoubtedly not up to appropriate standards. If it is to be

used I recommend using it to “sketch” an idea and then importing the model into ordinary Coot in order to fix details.

7.3.1 Nucleic acid painter

Aside from the protein backbone, there is another “chain” structure that is of fundamental importance in structural biology, which is the sugar-phosphate-nucleotide structure of DNA and RNA. I investigated the possibility of a similar “painter” tool, and though I did not finish it, I made some interesting progress.

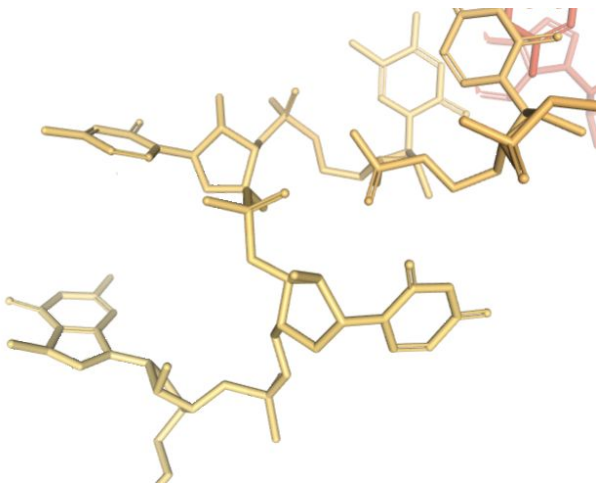


Figure 39: a heavily contorted RNA backbone that would be difficult to build[79]

The salient technical difference between protein and nucleic acid chains is that nucleic acid “rotamers” are very different from protein backbone rotamers. To think about complex protein structures as being broken up into simple pieces, the “phi-psi” formalism described above is reliable and has been established since the 1970s. There is an equivalent formalism for nucleic acid rotamers called “RNABC”, see figure 40. It is significantly more complex, and has more external degrees of freedom than the phi-psi formalism, and is less well-known. But it has utility for many structural biologists, having been implemented in structural biology software such as the Coot add-on RCrane[80].

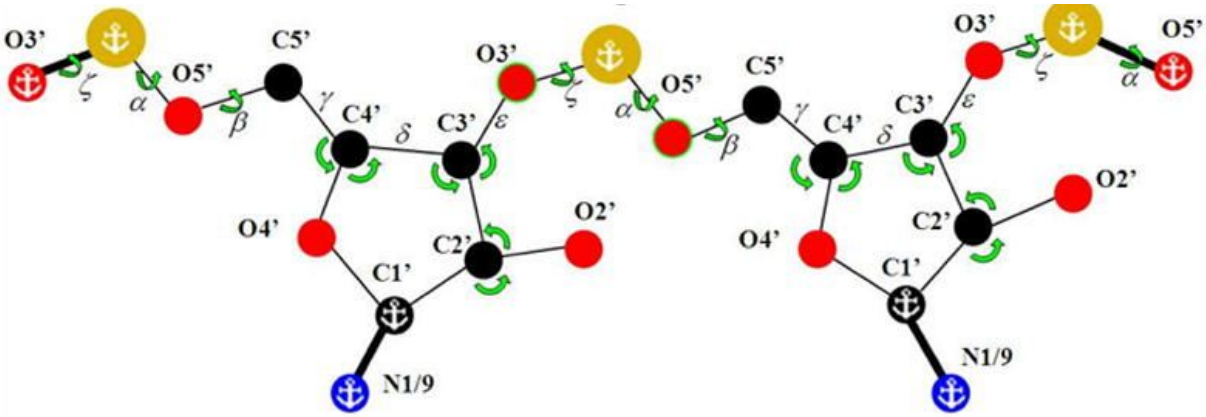


Figure 40: the formalism developed for computing nucleic acid rotamers, depicted and described in [80]. In this image, practically every angle between bonds is required to be a tetrahedral angle, and each green arrow represents a torsion angle. The atoms that have anchors on them are fixed in place (in 3 dimensions).

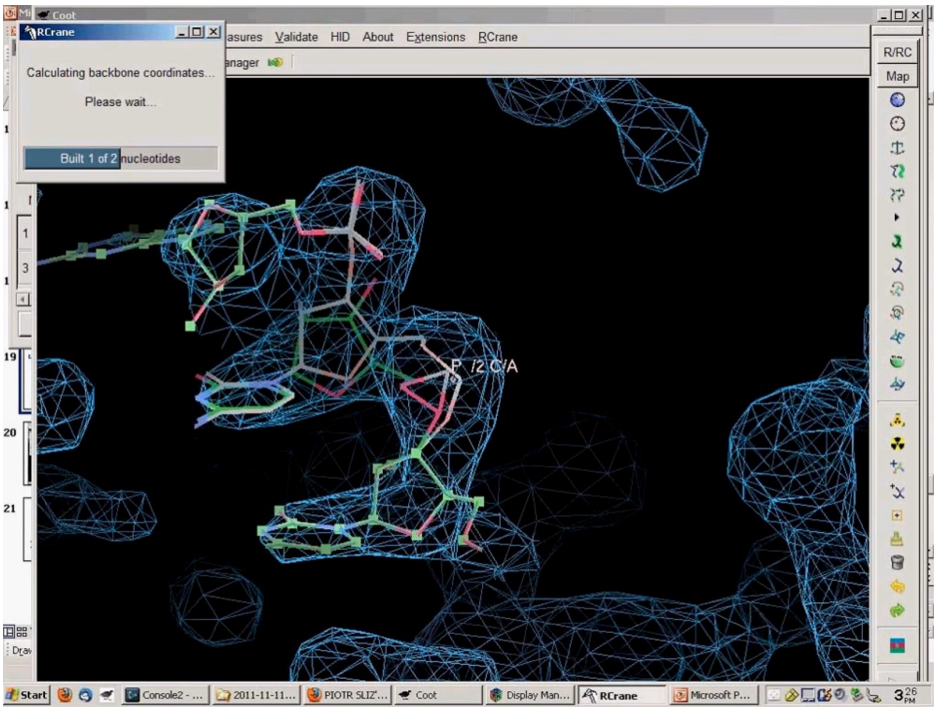


Figure 41: “RCrane”, a Coot add-on specifically for building nucleic acid backbones.

The way that RCrane implements the RNABC model is that the user picks a position for a phosphate; then they pick a position for the nucleotide it is connected to; then position the next phosphate; then the next nucleotide; and they finish with one more phosphate - and then RCrane will try to find a good fit for the atoms in-between (the sugars). This corresponds, in the diagram above, to laying down all of the anchored atoms and then having the torsion angles (and therefore positions and “pucker”) determined by the program. This can work, but has some

disadvantages - it is time consuming, especially if there is a desire to try out multiple conformations. Additionally, there are situations (for datasets with a resolution of about 3.2Å) where it is clear not only what position the phosphates are at, but also their orientation (i.e. the positions of all of their oxygen atoms, rather than just the single ones that are anchored in the far right and far left of figure 40). Submitting this situation to RCrane, there is a chance that it will ignore the indications as to the orientation of the phosphate.

It is possible that VR may be able to improve on some of these problems. In VR, one can specify a position and orientation for a phosphate and nucleotide very quickly, and change one's mind quickly too. I believe that this could allow users to very quickly explore a large number of possibilities by reorienting phosphates and seeing sugar rings change immediately.

Deciding *how* RNABC should be implemented in VR is a non-trivial question though, one on which I made some progress but not enough to implement it. One method would be very similar to RCrane as described above, but it might be better, for example, for the user to grab a bond in the pentose ring rather than a phosphate.

7.4 Extensions to nonlinear chemical topology

Proteins, as stated above, have a simple fundamental topology that makes it easy to plan and interact with them - they are a linear chain. The same is essentially true of nucleic acids, the pentose sugar making a slight difference in that it has chemical constraints coming from three different directions. In any case, both are well-studied and simple in the sense of not having many free parameters.

In theory a more general and more powerful framework is possible: a tool for allowing the user to define and even automate their own constraints. This would encompass both of the above, and much more, including "backrub rotamer"[81,82] and "sugar pucker correlate"[81] control, and dealing with complex drugs and sheet-like molecules such as graphene. It could work as follows:

1. The user is looking at an arbitrary molecule, potentially a sheet or even semi-constrained lattice
2. They grab an atom. This causes all chemical constraints on it to be visualized. In particular a minimal set of atoms whose positions influence its constraints become highlighted; all bonds of specified length become highlighted; and all bond angles of necessary size become highlighted.
3. They may move their hand, and the atom may move as much as possible while keeping to the constraints. This may mean not moving at all, or it may mean flicking between two points, as in the case of a backbone nitrogen.
4. The user may relax certain constraints, for example by tapping one of the "anchored" atoms allowing it to move any way at all, or to be less extreme, by flagging it to become part of a rigid body. The first option is possible in Coot, but not the second.

5. The user can “save out” a certain set of designations for constrained and unconstrained atoms. This may allow them to make their own version of the protein painter tool. This is the approach used by The Geometer’s Sketchpad for drawing parameterized shapes.[83]

8 Selective visibility

Proteins are intricate three-dimensional shapes, and model building involves inspecting every single part of them very closely, including their interior. Even once it has been decided how the model and map are to be visualized (see below), the user must be empowered to quickly choose which parts they want to see, because they do not want to see the whole thing.

This chapter's discussion is largely independent of the fact that molecular graphics is the focus of this thesis. In principle the same arguments and conclusions should apply to other 3D modelling programs, such as those used for architecture and inspecting MRI scans.

8.1 Selective visibility in Coot

In the context of Coot, selective visibility is accomplished by controlling the "clipping planes" or the "clipping slab", see figure 42. Changing the clipping plane depth is a command bound to the "d" and "f" keys - these are keys that sit directly beneath the resting position of the left hand, which shows how commonly they are used. In the 3D world of everyday life, it is obviously quite common that a person would want to look at something behind or in front of what they are currently looking at. On a 2D screen this has to happen quite slowly.

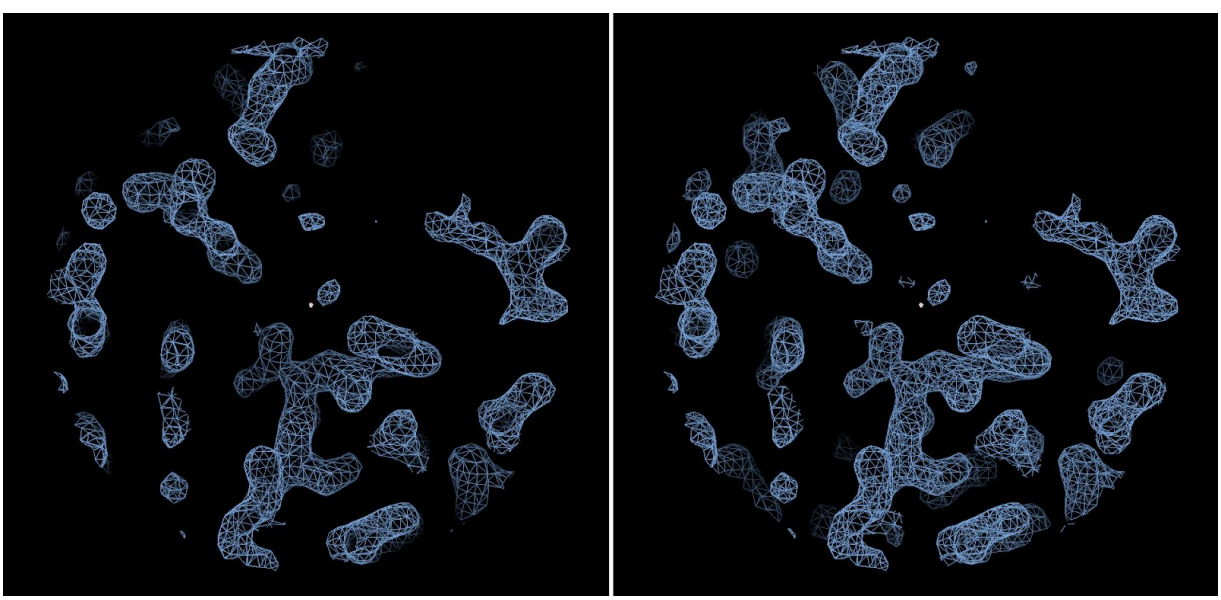


Figure 42: a map displayed in Coot with different values of "Clipping plane depth". On the right, the clipping planes are further apart, so one can see more of the map extending backwards and forwards.

Figure 42 depicts what is being talking about. There are some areas in the data, of course, where there is no density. Looking *behind* the density, I also see empty space, but of a different

kind - there may well be density there, but it is not shown because it is outside of the clipping slab.

In principle a beginner could make the mistake that the empty space behind the visible density means that there is no density there. Two things help avoid this mistake:

1. The user is expected to constantly rotate the view, putting different areas into and out of the clipping slab, giving a clear sense of what else one might not be seeing.
2. The “fading” or “fog”, which indicates the direction and distance that unseen things may lie in.

8.2 Selective visibility in CootVR

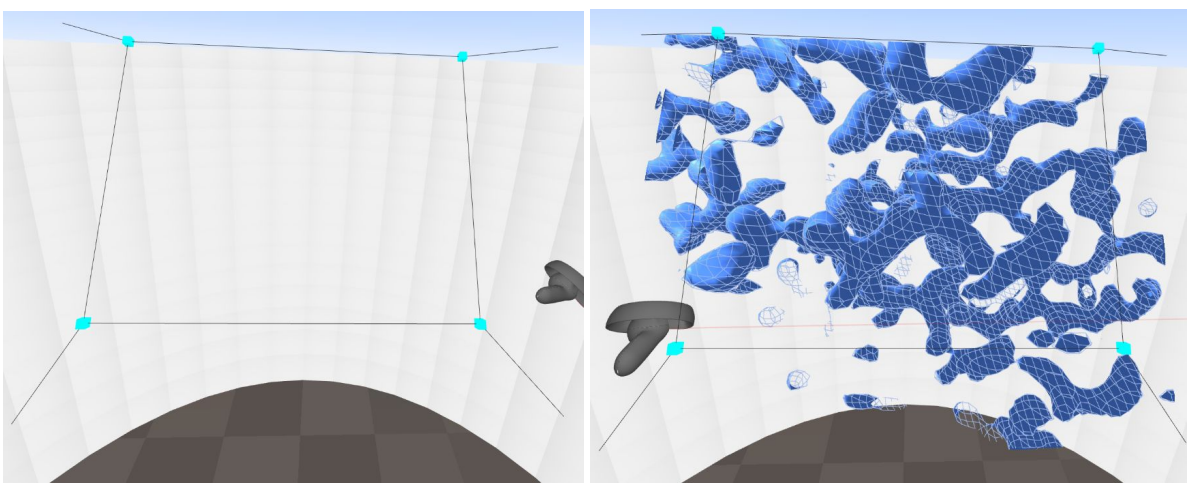


Figure 43: the “visibox”, without (left) and with (right) something inside it.

Our approach is depicted in figure 43. The molecule and map are visible within a specific 3D area that is enclosed by line segments. The area is in the shape of a “frustum”, essentially a square-based pyramid that is cut off at the top. The peak of the pyramid is the place where the user puts their eyes, i.e. the part that they look at is the flattened top of the pyramid. In the context of Coot and other molecular graphics programs this top is called the “front clipping plane”. In principle the plane could be curved or skewed, but it is simplest, and therefore probably best, for it to be a flat plane.

At the corners are small blue cubes which can be grabbed and moved if the user wants to change the horizontal, vertical, or lateral size of the volume. If a blue cube is grabbed and moved left or right, the cube above or below it will move with it. Additionally, the cube that is horizontally across from it will *mirror* its movement, maintaining a vertical line of symmetry down the middle (we cannot think of any motivation that the user would have to desire an asymmetric volume). If the cube is moved towards, or away from, the place where the user’s face is expected to be, all the cubes will move the same way. This automated movement is very useful; a sophisticated change to the visibox can be made with a single hand movement.

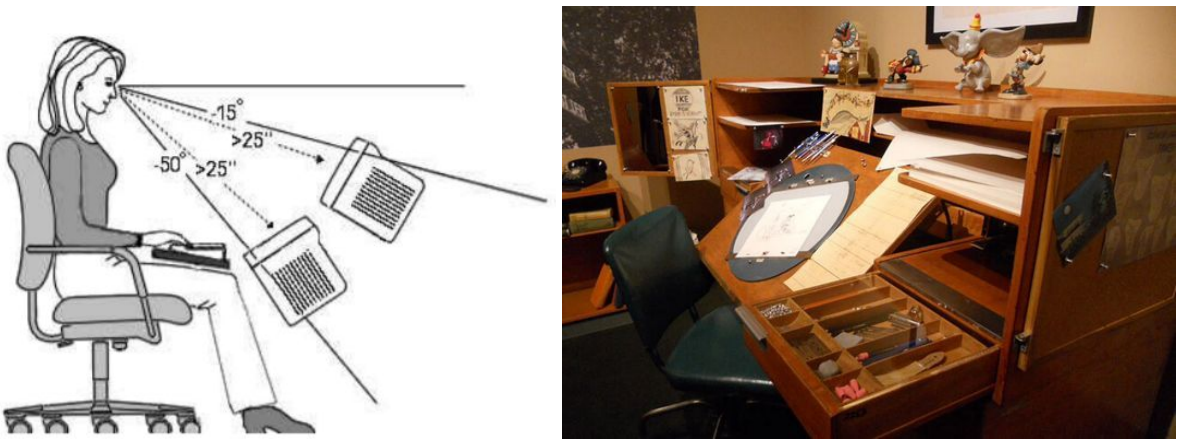


Figure 44: Left: image from “visual ergonomics at the office”[84]. Right: artist’s desks[85] are almost always tilted towards the expected location of their head.

With regard to its placement, the volume is tilted up, towards the user, in accordance with well-understood ergonomics, see figure 44. Originally it was directly in front of the user with front clipping plane facing directly forwards. This was very uncomfortable for the neck, and I felt the constant need to adjust it; I usually ended up with the top clipping plane brought down quite low so that I was essentially looking at the top edge, which is visually confusing. Tilting the pyramid up solves this problem.

There is widespread public concern that working with computer monitors can cause people to become “hunched”. Consequently many monitors on the market are designed to have their center at eye level and cannot be tilted. In actual fact there is no evidence that this has any health benefits.

8.3 Alternatives experimented with

Previously, I gave the user the ability to pick up the volume and move it around, or even rotate it, arbitrarily, independent of the molecule. While this was a visually-intriguing thing to do, I removed this functionality on the basis that it was also making the user explore a much higher-dimensional space of options than they needed to (see the discussion of “dimensionality reduction” in the conclusion).

Figure 45: it is common in spelunking to have a torch mounted to one's head, which makes it so that the only things one can see are things that one is directing one's head towards.

One idea for automatic visibox movement, was experimented with, where visibox was essentially a slab attached to the user's head. This meant that they would have a small area of clipping volume in front of their face at all times whose size they could control. Automation is in general good, of course, and this seemed like a simple re-interpretation of the Coot approach in VR: in Coot, the molecule stays in place and the user moves around it, and the clipping slab with them. I also believed that it would feel intuitive, because it would be similar to a head-mounted torch, figure 45.

However, this was incredibly unpleasant and confusing. This was partly due to constant distracting flickering as atoms entered and left the viewing plane. It was also because one would see something, then move one's head, and then forget exactly where it was when one looked back. I also considered having the visibox always-mounted on the user's left hand, like holding a lantern or candle. This idea was dismissed for the same reason. Both of these ideas are, I would say, examples of the ”realism fallacy” described in the conclusion.

I considered giving the volume the simpler cross-sectional shape of a circle (which would make the whole thing a cone), but the cubes being on corners allow for this simple way of changing the shape and size. I also, originally, had the shape as a rectangular prism (i.e. with the side, top, and bottom clipping planes being parallel with one another and at a right angle to the front clipping plane). This is problematic though, because it means that if the user moves the model to the left, then they will see a lot of atoms pop in and out of visibility towards the back. With the frustum shape, this problem is eliminated, and more can be seen. To attempt to completely eliminate flickering, I at one point implemented a kind of “hood”, figure 46, that would obscure all atoms except those seen directly, but I believe that this increased visual distraction and so removed it.

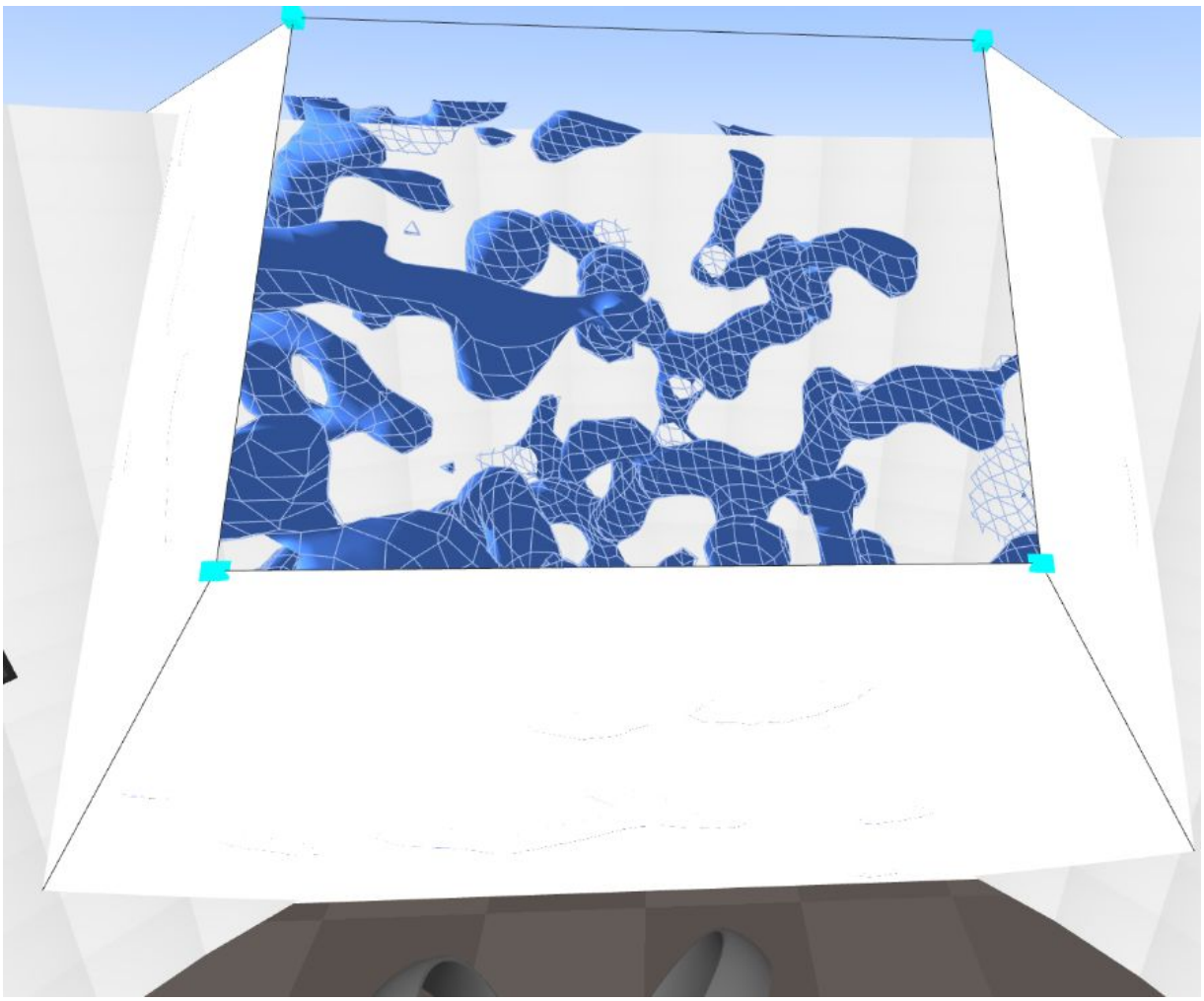


Figure 46: the “hood” on the visibox, which made it so that when the molecule was moved one would not see atoms appearing and disappearing on the sides.

8.4 Full visibility

Within the VR industry, discourse around “the brain as a 3D computer”[48] is common. What this particular statement means is vague, but some developers (including us at the beginning of this project) work under the impression that a human can fully absorb any amount of information about 3D structure and memorize it, so long as they are physically *inside* the molecule or “the molecule is their environment[14].

Having the molecule literally be the user’s environment is equivalent to asking what happens without any clipping planes. The answer to this is that at all times there are is an atom or bond mere centimeters from the user’s eyes which obscure their view (and makes it hard to focus their eyes too). If one tries to move these away from one’s face, it is probable that more will still be more atoms in front of them. This plainly makes it harder to understand the molecule’s 3D shape, not easier. Of course, it is worth trying to “think outside the box” and look for ways that

new technologies are transformative. But claims made about how a protein can be understood in a new way when thought of as an environment are very nonspecific about what the “new understanding” allows one to do.

9 The panel

CootVR contains a specific object called the “panel”, figure 47. It is a unmoving, curved surface on which many other 2D objects sit. The function it fulfills is loosely comparable to the function that control panels and screens have in real life.

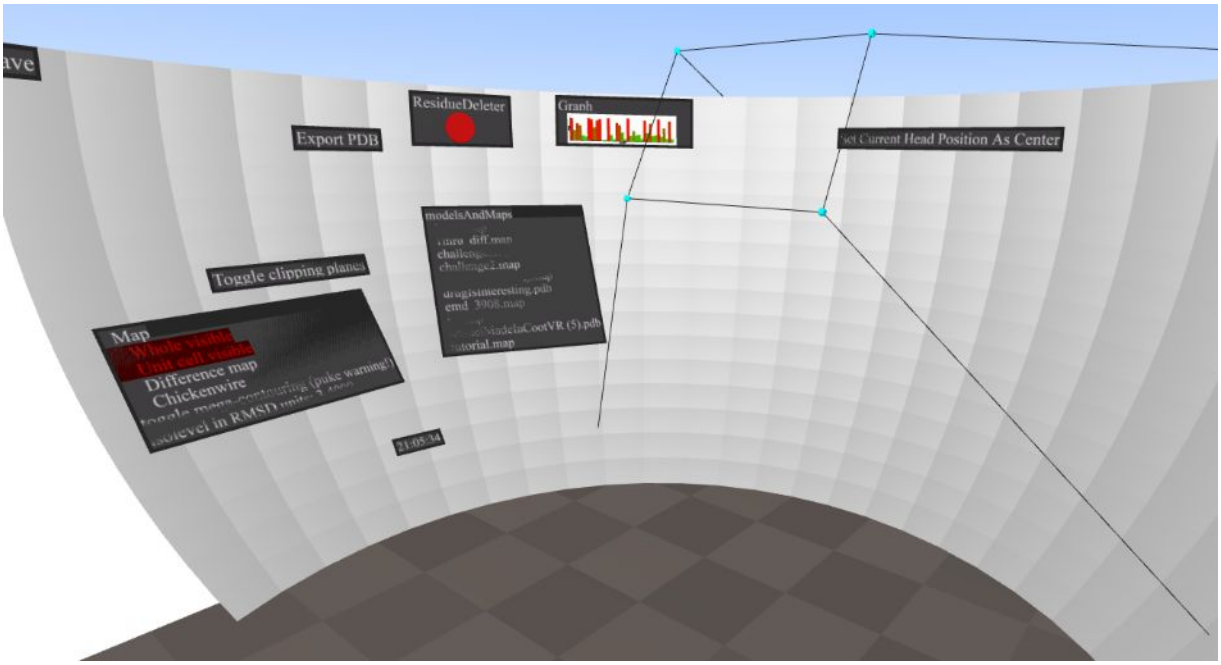


Figure 47: the panel, from a perspective adjacent to where the user actually sits.

There are a few different sorts of objects that can be placed on the panel:

1. Non-interactive information, for example, what time of day it is.
2. Realtime-updating graphics, for example a graph of the ramachandran values of the amino acid nearest the user’s hand
3. Interactive buttons, such as the “export pdb” button
4. Tables, such as tables of binary options or lists of files in a directory that the user can “click on” in order to load
5. “Tools” that are to be “picked up”, see other sections
6. A fast way to navigate the model, such as the sequence view

In normal Coot, all of these functions would have their own window or button on the interface, following the “windows, icons, menus, pointers” design pattern[86]. The panel is very similar to this, the only real change being the curvature and the presence of more room, which means that menus do not have to be “folded away”.

Figure 48: The interface for Oculus Home, which is also essentially 2D

Figure 49: United States military academy west point cadet chapel organ.

The panel's shape and design is modelled on organs (for example figure 49) and cockpits (we considered having an overhead panel of buttons similar to a cockpit, but the need did not arise, and this would entail neck-craning). Its top edge is at eye level, following figure 44. There are some other apps that use something like our panel, for example Oculus Home, figure 48. Many VR programs have all of their controls and menus appear on the wrist, for example Tilt Brush, figure 50. I believe this is a mistake, as it takes a longer time to perform a tool selection; it requires both hands for selection; and it requires things to be small, i.e. not verbally labelled.

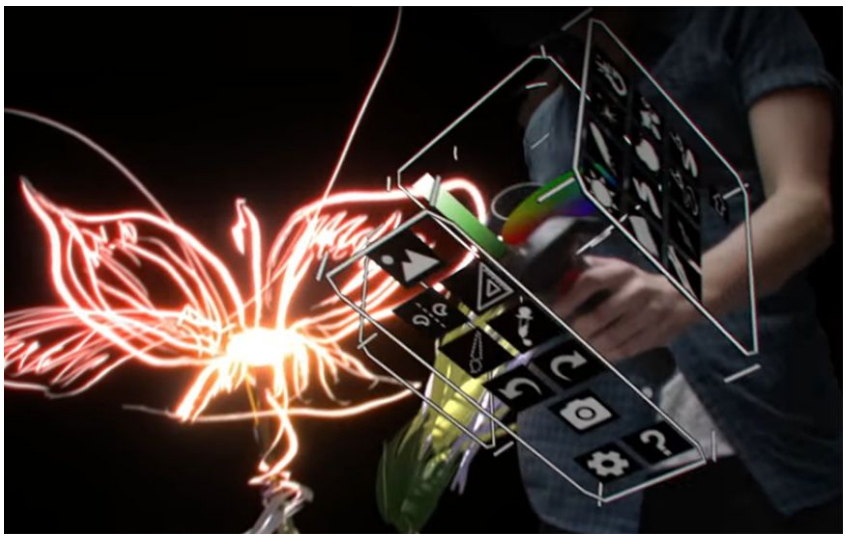


Figure 50: In the VR drawing program “Tilt Brush”, the controls all sit on one of the hands

Many VR interfaces, including that of Tilt brush and Oculus Home, have gaps between the windows. This is because these apps are intended for entertainment, and the gaps useful for being impressive. Typical UI elements are, by design, uninteresting and functional to look at. By having the cracks between them, the user is able to see more interesting things behind them (Tilt Brush takes place in a pleasant field during twilight, for example. I decided against this approach; the users of CootVR have a more important job to do, so there is no distracting background.

I have set it up such that “tools” go on the right, and “metrics” and “options” go on the left, but the user can change this by “grabbing” the “windows” and moving them. I expect this to be a common part of the workflow, because lesser used windows will be far away from the central position, but when the time comes to use them, it is good to drag them to the center.

The panel is ellipsoidal, which makes it so that practically every point on it is equidistant from the user’s eyes, and also makes it so that options that are lower down are tilted towards the user’s eyes. The assumption of a “default” or “encouraged” user head position is controversial in the VR community[87], because it discourages the user from moving around the space. But I would argue that no benefit, for CootVR at least, comes about from moving around the space, see the section on “software as environment”. With this said, it would be bad if the user had to keep their head *exactly* in one position in the physical world, and for that reason I created a button that recenters the panel on whatever the current head position is.

9.1 Hand interaction

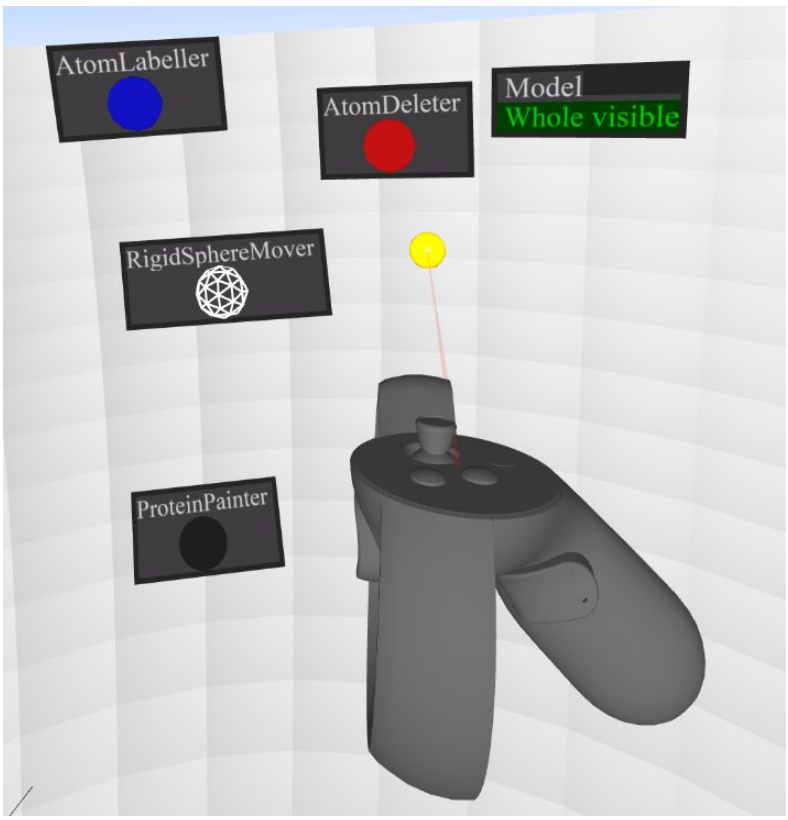


Figure 51: selection of objects on the panel. The yellow point moves wherever the red laser coming from the controller is pointed, and the rectangles can be grabbed and moved too.

The user has a pair (one per hand) of “cursors” that sit on the panel, and can be used to “click on” the objects that sit on the panel, see figure 51. If the user puts the yellow ball over the proteinPainter, say, and presses the trigger button, the tool will teleport into their hand until they press the trigger button again.

Originally, these tools were floating in mid-air and the user would pick them up by overlapping their hand with them and pressing the button. But this requires putting one’s hand in the correct x, y and z position, which is three degrees of freedom, whereas to pick up something they can see the user ought to only be choosing on two degree of freedom (because our sight is fundamentally 2-dimensional, see above). There are objects that float in space, but only a few: the corners of the “visibox”, see above; a “lightbulb” that can be repositioned in 3D space to change lighting; and the molecule itself, all of which are picked up with a different button from the panel-pickup button. So it makes sense for the interface (the panel) to be 2D in general[88]; this is discussed further in the section on dimensionality reduction below.

Note that the lasers come out of the side of the hand rather than the front - I find that this works well because the front is where the model always sits. It is, perhaps, surprisingly easy to get used to pointing with the back of one's hand. There is some "smoothing" on the movement of the cursors (the same is true of computer mice); this smoothing makes it so that the cursors are not affected by millisecond-scale jitters of the user's hands, which makes it easier to aim the cursor. One might expect this to feel uncomfortable, but it is barely noticeable at all - in fact the jitter that occurs without smoothing feels as though it is coming from the device rather than the user, even though it is definitely coming from the user. This is similar to how desk surfaces and mousemats provide some "resistance" to mouse movement, allowing the holder of a mouse to make very fine adjustments to its position.

The user is not actually intended to ever touch the panel or get close to it, because it is not necessary. To make it more comfortable to look at (see "vergence problem" above), the panel is placed 1.6 meters from the expected location of the user's head, too far away for anyone to touch. This goes against early VR UI design guidelines - at that point it was believed[89] that the user would want to touch everything, essentially because touching things is fun.

Figure 52: a carpenter's tool belt

Another idea for tool selection I originally had was a "utility belt" similar to those worn by carpenters or superheroes, see figure 52. Intuitively it seemed like a good model, since there are a variety of tools that the user wants in their hand. However, in VR users have an advantage that carpenters do not, which is that it is not necessary to perform time-consuming movements around the location they are working, and there is not the force of gravity. Therefore, I can have the equivalent of a toolshelf that is always in a singular highly accessible location - that is the use of the panel.

One more ambitious idea for the panel was to try to make it possible for the user to use it *without looking at it*. This would be accomplished using carefully designed sound effects. The idea was that they could keep their gaze fixed on the molecule, and point their hand at something they could not see, but instead they would listening out for it to snap to the correct option. Then they would press the trigger button and obtain the correct tool in their hand, and

bring their hand back into their field of vision, without having moved their gaze at all, just as a carpenter might do with a tool on their toolbelt. This may well not work though, so I did not pursue the idea very far.

10 Visualization

In this section I describe how I tackled the main visualization questions and technical details.

As a general point, many modern 3D applications use the “engines” called Unity and Unreal (which fortunately do not incur licensing issues, as they are free for non-commercial work). Early work on this project was based in Unreal, but I came to the conclusion that, at the time at least, Unreal was more intended for video games than our application. For example, in Unreal applications, the user cannot usually import an arbitrary file at runtime.

Instead I decided to use “three.js” and “WebGL”, APIs that would allow us to create a 3D application running in the browser. This has the advantage that the user can access the latest version of the software instantaneously, and from any machine (there is no “installation” procedure).

WebGL has the disadvantage that it does not run as fast as what is possible with Unreal or Unity (or OpenGL). However, for the most part this has not actually affected our project, because the applications that Unreal and Unity are optimized for is loading in large quantities of 3D data (an order of magnitude more than what structural biologists work with), so I have not experienced any performance problems.

10.1 Visualization and rendering of scalar field

Visualization of data of interest to structural biologists is non-trivial; the datasets are conceptually quite exotic, in the sense that it does not have a particularly familiar analogy from everyday life (unlike, for example, data that simply counts the number of occurrences of an event over time). For the task of model building, which is geometric, datasets are mostly thought of as a 3D “scalar field” (the raw datasets more complicated, see above, but for our purposes this is a very reasonable starting point). I will describe what a scalar field is with a number of analogies; each one is useful, because our task is to find the visual representation method that is most simple, visually and conceptually.

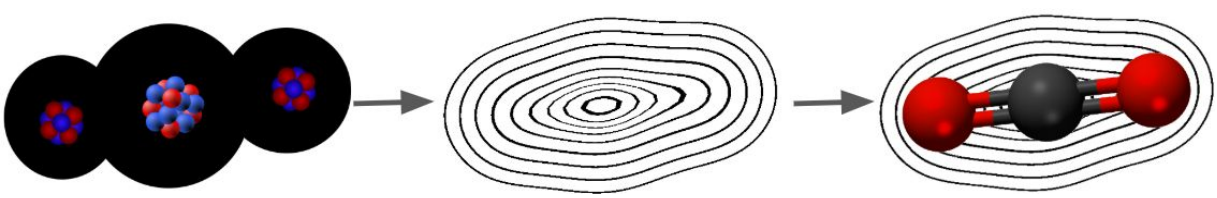


Figure 53: the basic idea of the kind of data that CootVR deals with. On the left is the “actual” molecule, a carbon in the center with two oxygens on its sides; in the center is the (high resolution) data one might obtain from the molecule; on the right is the model one would hope to build of the molecule, where the atoms have been placed in their model where one might suspect they are - in this case one would be exactly correct.

A good verbal analogy for an electron density map is a “blurry” three-dimensional picture, or possibly a “cloud of smoke”, where our dataset tells us the smoke’s density at every point. See figure 54 for an intuitive representation of the data as it comes to us.

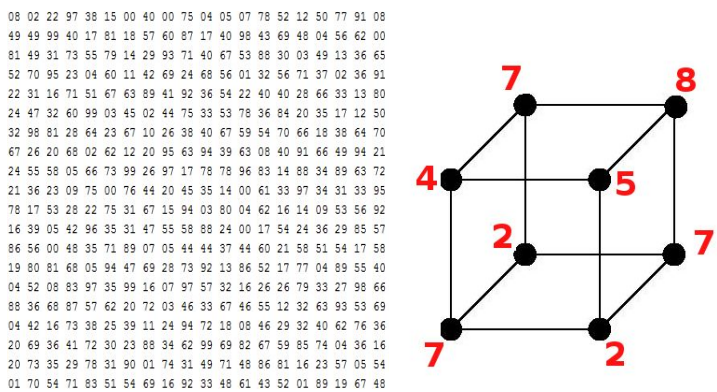


Figure 54: Left: a 2D scalar field dataset. Right: the tiniest-possible 3D scalar field dataset; if one imagines many cubes like it stacked to fill space, that is a more typical 3D scalar field. The number at a grid square is the density of “smoke” at that location. Two caveats to it are that the dataset that is obtained is a three-dimensional grid, and that it sometimes comes on a non-orthonormal grid, and so needs to receive a trivial scaling and shearing.

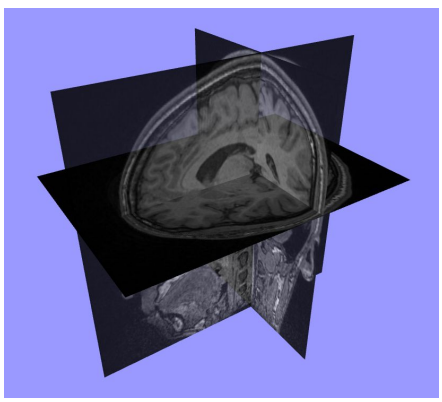


Figure 55: From [90], an MRI scan is an example of a 3D scalar field. For a human to “take in” a full MRI dataset, they usually have to view multiple “slices”; it is impossible to “see” or even imagine the whole thing at once.

Data like this is difficult for a human to deal with because it is intrinsically three-dimensional, i.e. it can have a very complicated “interior”, see figure 65. Humans generally appear think about

the world in a way that is mostly two dimensional, which I supplement with “tricks” to make ourselves do small pieces of three-dimensional thinking. A concrete example of this is the fact that our retinas are flat, and so can only receive, in a sense, a 2D array of data. Most computer displays, including VR headsets, only display a 2D array of pixels too.

It is a very basic necessity to visualize the user’s data though. I experimented with several different ways, as described below: slicing, volumetric ray casting, and contouring.

10.1.1 Contouring

Contouring a 3D scalar field involves picking an arbitrary value (“cutoff”) and rendering a single 2D surface such that the value of the scalar field at every point on that surface is equal to the cutoff. For electron density datasets and anything like them, this creates surfaces in 3D.

An analogy is contour lines on a map, see figure 56. The difference between a contour line on a flat map and a contour “surface” for a 3D scalar field is that the contour surface usually has only a single surface rendered, instead of multiple concentric surfaces (although an interesting exception to this is described below).

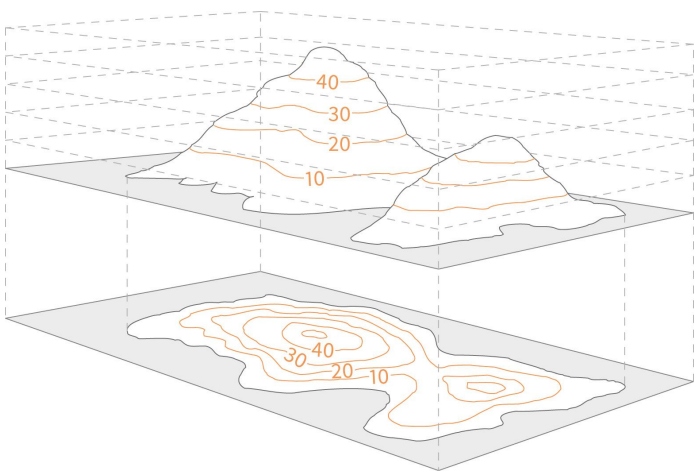


Figure 56: contour “lines” (analogous to a contour surface) allow a 2D scalar field, representing the distance above sea level of a point on a landscape, to be depicted in 2D. Figure from [91].

Almost every program that draws contour surfaces, including ours, allows the user to specify the cutoff value (“isolevel”) and then that set of surfaces. For the purposes of technical discussion exact value of the cutoff is therefore considered user-specified.

Contour surfaces are a conceptually simple and powerful way to visualize the scalar field, allowing potentially very complex scalar fields to be turned into sets of geometric shapes that can be understood at a glance. The user is given has the natural sense of what is “inside” and “outside” a surface, which corresponds to the fact that on one side of the surface, the value of the scalar field every point will be higher than the cutoff, and every point outside of it will be

lower. This affords a common strategy: if the user sets the cutoff appropriately, *atoms inside the surface is a good thing*, and *outside is bad*.

The major disadvantage of the contour surface approach is that some information is necessarily missing. The analogy of a cartographic map, figure 56 illustrates this. When contour lines are seen on a a map of an area, the curves can indicate practically all the relevant height information. However, when looking at a single contour surface, some information is necessarily missing, in the same way that a great deal of height information would be missing if only one contour line was present on a map.

Another way that a contour surface can be misleading is that there are subtle ambiguities involved in the question of how it ought to appear. To render the contour surface, as discussed below, I assume that the surface is completely continuous, and some “interpolation” is necessarily involved, i.e. “details are filled in”, specifically in the form of triangles. This compares unfavourably with volumetric ray casting, also described below, where a given scalar field viewed from a given perspective has a single well-defined appearance, and interpolation is less misleading.

10.1.1.1 Implementation

The visualization of the contour surface is very integral to the software, so it is worthwhile to discuss certain choices made in its implementation.

Processing the scalar field is quite computationally demanding. Our goal was to make it very fast - fast enough to keep the 90fps limit to avoid nausea described above, and also fast enough that the user could quickly “search” through different contour levels to choose the one they wanted. I therefore chose a particular schedule for the algorithm to run on, with the following properties:

1. I have a separate thread that does most of the processing;
2. It does it only when necessary. This means that there can be slowdown, but it is temporary. This also means that it is done on the CPU (and is saved to an array) rather than the GPU
3. It works on small pieces. This means that the user can be changing the contour level and can see it update instantaneously.

Rendering a contour surface based on a scalar field is a fairly common task in computer graphics, and so there is a classic CPU algorithm for it known as “marching cubes”.

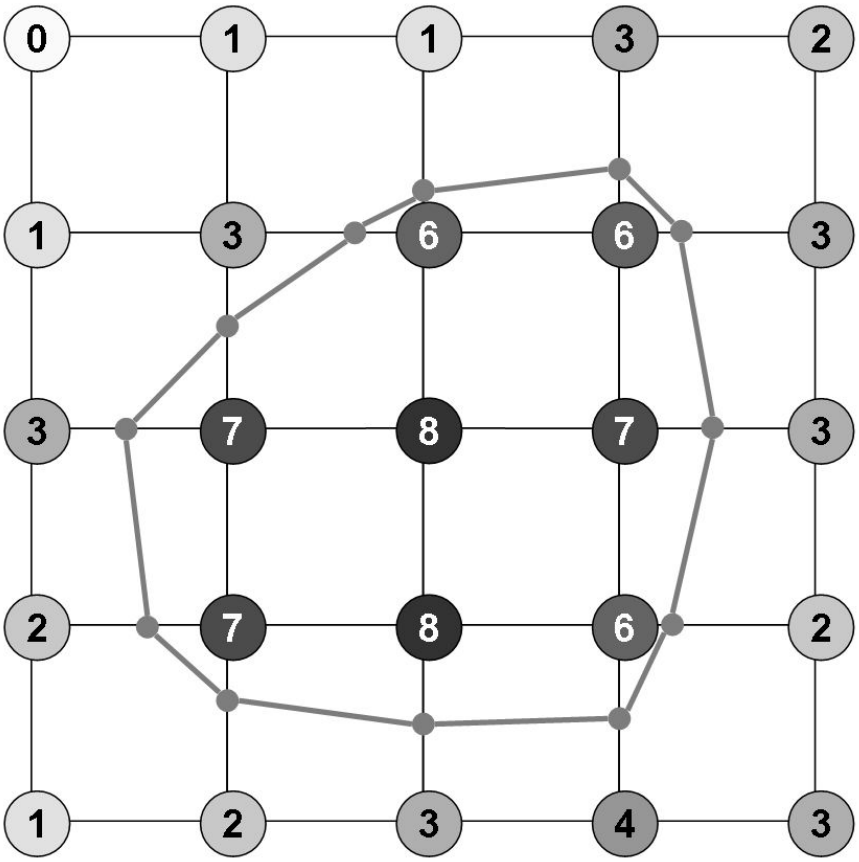


Figure 57: A lower-dimensional analogy. A contour line is drawn through a scalar field with a isolevel of 5. The numbers represent scalar values at their respective grid points

Consider attempting to obtain the contour line in figure 57 based on its grid values. To do this, I iterate through the squares. For each, it is decided if there is a line that needs to be drawn in the square, which is based on the binary questions of whether the values at their corners are above or below the number 5 (the “isolevel”). For example, nothing needs to be drawn in the top left grid square, because all of its values are below 5. Nothing needs to be drawn in the three squares inside the contour line either, because their corner values are all above 5. But for both squares in the center of the right side, a line needs to be drawn cutting the top and the bottom.

If a line is to be drawn, the exact points for its beginning and end need to be determined; both will be somewhere on the two sides of the square that are “piercing” the contour shape. This is equivalent to saying “where, along the line between the two grid values, is the value of the scalar field equal to 5 (the isolevel)?”. In the case of figure 15, and in our implementation, these points are determined by making the unrealistic but useful assumption that the scalar field is changing linearly along the length of the cube/square’s edge. For example, the central horizontal edge at the left of figure 15 has the values 3 and 7 on it. Therefore, the algorithm proposes, if I start at the place where I know that the electron density value is 3, if I move 1/4th

of the way towards the place where the electron density value is 7, I will be in the place where the value of the scalar field *would be 5, if it had been measured there*.

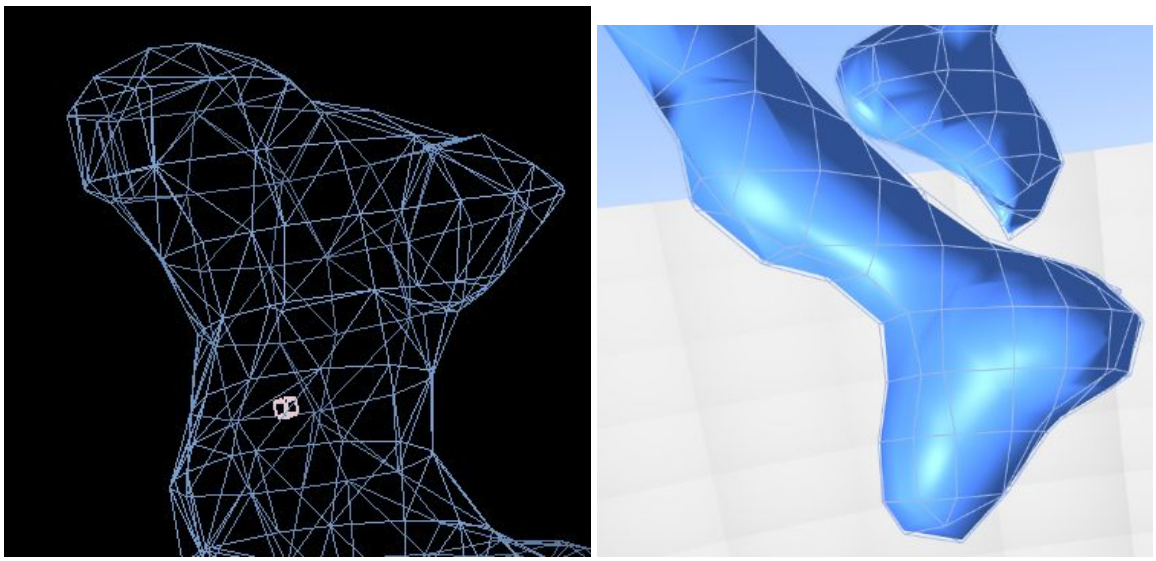


Figure 58: Coot's rendering of a scalar field, the 3D equivalent of figure 57

The method above is called "trilinear interpolation" when applied to a 3D scalar field, and is used by Coot, figure 58. However, the assumption that the scalar field changes linearly between the grid points is not at all justified. Consequently, the result is jagged, with sharp turns at every corner. This is arguably a problem, because even though I do not know what an electron density map is shaped like at non-gridpoint locations, I can be certain that it is not jagged.

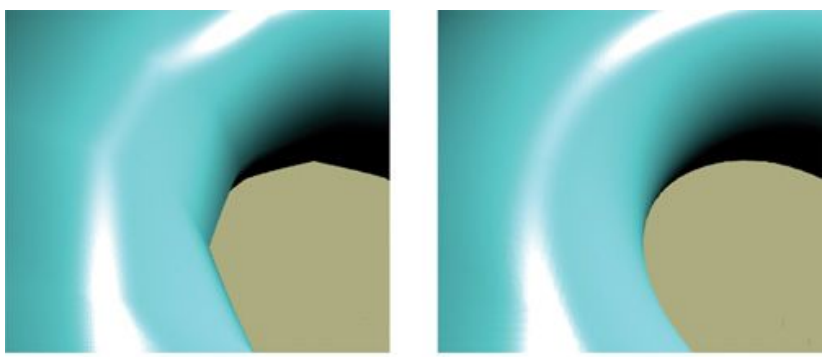


Figure 59: a closeup of a scalar field with a contour level drawn using two different kinds of interpolation.

There is an alternative to trilinear interpolation called tricubic interpolation, shown in figure 59 compared with the same data under trilinear interpolation. Tricubic interpolation looks better and reflects the fact that the real data would never be "jagged". However, one can argue that at this point the structural biology community is very much used to jagged surfaces, and that it may even be a positive thing giving some indication of the "limits" of the data.

10.1.1.2 Details

Initially I copied Coot's style of surface presentation, the famous "chickenwire" of figure 58, however I found this to not be ideal. In Coot, it is satisfactory to have the "slab" (see "selective visibility" above) be quite thin, and only really be looking at a few shapes within density. In VR there is an intuitive feeling that the slab to be about as thick as it is tall and wide, possibly because of the presence of stereoscopic vision, or possibly so that when it is rotated it needn't change much. In any case, if an approximately-spherical chunk of density is rendered in the "chickenwire" style, it is very difficult to see what is behind what, see figure 60.

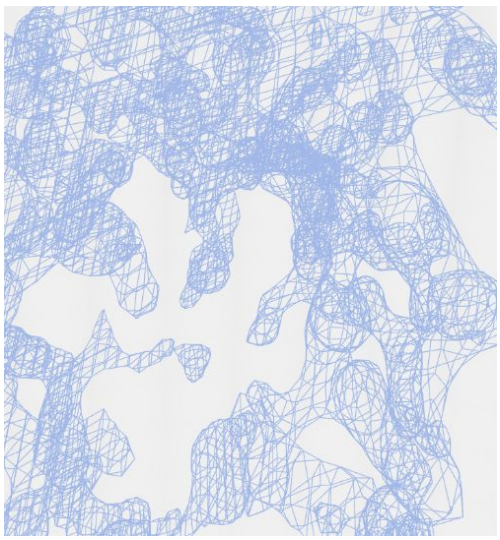


Figure 60: "chickenwire only" block

Therefore, our representation is a combination of a chickenwire "front" and a solid "back". I originally had a transparent front too, but found that it added little. I ensured that the curvature of the solid surface was clear, which requires calculating a normal map - this is computationally somewhat costly, but I believe it is worth it. This representation has attracted positive comments in the structural biology community. The good thing about it is that it allows the user to glance at a large number of atoms and very clearly see which ones are "inside" the surface.

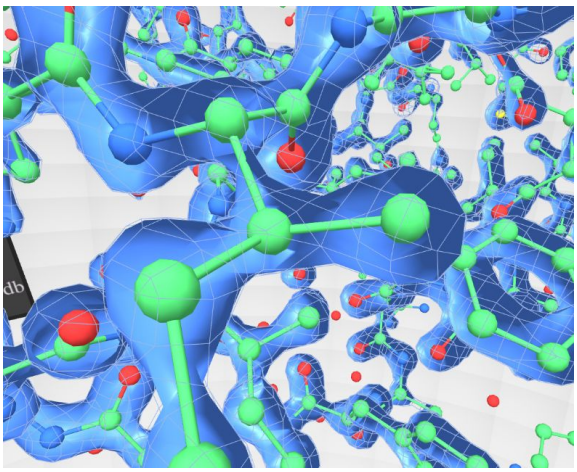


Figure 61: the contour surface around a model in CootVR

This surface object, as perceived, is not very much like any object from everyday life. It is somewhat like having hollow object made of one-way-glass, i.e. when it is looked at from different angles one always only sees the “back” of it. This is obviously not something that humans evolved to be able to inspect, so it is interesting how well it works. I note that seeing the atoms in place with surface behind them is somewhat like looking at an object “nestled” in a curved “palm”, which I think aids 3D intuition.

The inability to see anything “behind” the surface is, I claim, a blessing that makes the image less cluttered, though it has been criticized because in comparison with chickenwire the user can see fewer things in the background.

10.1.2 Volumetric ray casting

“Volumetric ray casting”, or simply “volume rendering” (“VRC”) is a visualization method for scalar fields that is most faithful to reality, at the cost of being at least more confusing to look at. It involves taking the “smoke” analogy literally, rendering the dataset as a shape in 3D space where the high-electron-density regions of the interior are opaque, and low-electron-density ones are almost completely transparent.

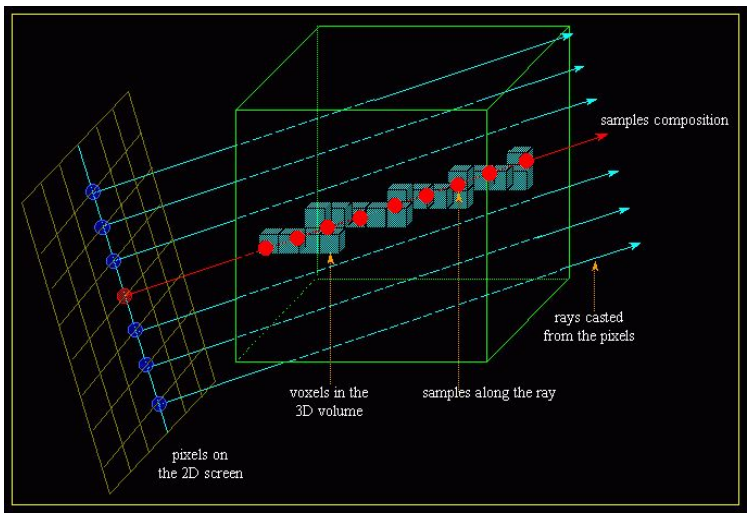


Figure 62: light rays are “cast” through the data, accumulating color values as they go

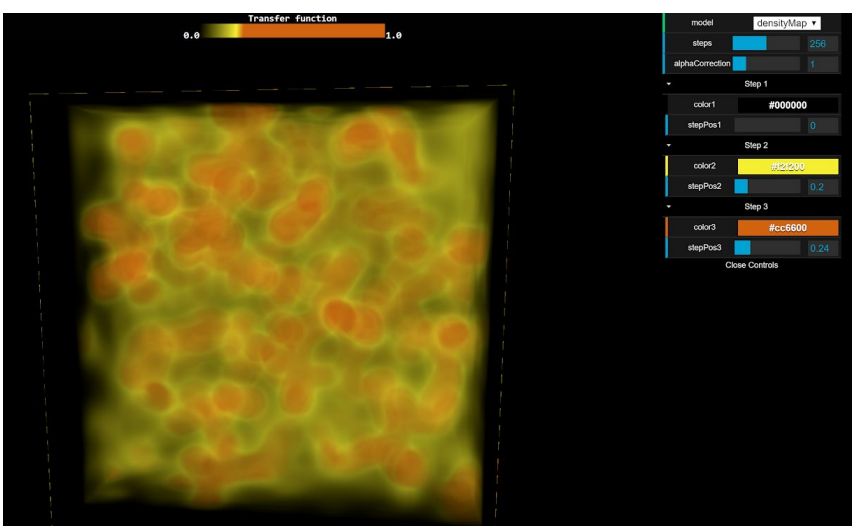


Figure 63: an electron density map visualized with volumetric raycasting, available at http://hamishtodd1.github.io/VRC [92], and based on [93]. The interface on the right allows the user to control precisely how density level of a voxel affect a light ray cast through it.

Consider figure 13, our implementation. Looking at a given place in the dataset, one sees through layers of hazy density, behind which I can discern more density. Possibly, far back or up close, I will see a solid area, i.e. a region of high density, which fully obscures everything behind it. With all the haze adding up, one can “see” all the data at every point that is being looking at. This has the powerful advantage, in comparison with contouring, that the user need never change contour level - all information from all contour levels is represented.

It turns out that this method is impractical. The volume, figure 63, is confusing to look at - even when one is rotating it. It *can* be interpreted, but for us at least, it is essentially only interpretable when our sight latches onto a specific “essentially solid” part - i.e. a place in the haze where one

can consider a “cutoff”. Worse, if the density is particularly vague, one can end up giving up on thinking seeing “depth” and treating a what one is seeing as two-dimensional. One could try to argue that this is as it should be, that if the user is going to try to find solid things in the field, it ought to “feel wrong”, because solid objects are not there. But, as discussed in the section on contouring, it is reasonable for the user to want help with this simplification.

The reason that this approach fails is probably because the human visual system evolved to look for and process information on solid objects, not smoke. If it had been otherwise, volumetric ray casting might have been a better value proposition in this domain. Volume ray casting may make more sense in areas such as anatomy scans, where the scalar field essentially has several discrete values, eg one value for bone, one for muscle, one for arteries. However, for electron density, which is a homogeneous material, it is just like looking at smoke, which not preferable to looking at a solid object.

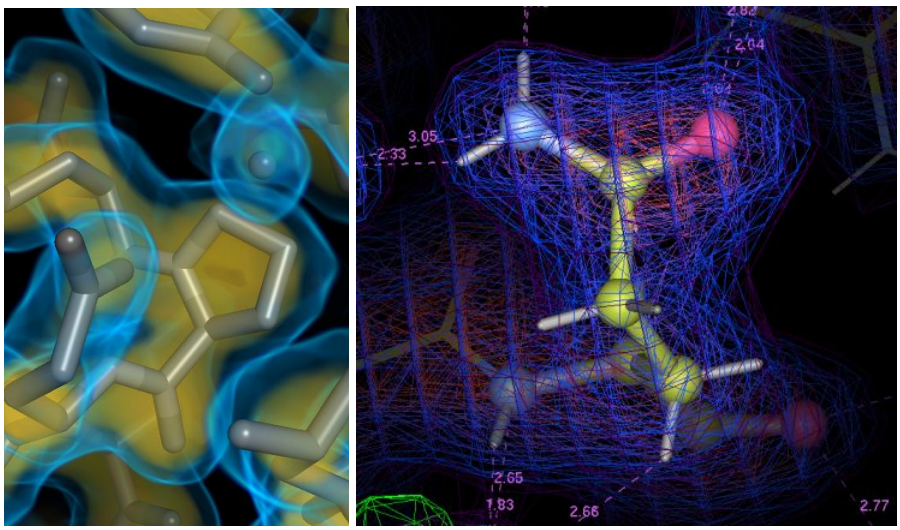


Figure 64: Left: a way of visualizing electron density which is functionally similar to volumetric ray casting, from[94]; right: the same method used in Coot, which is supported by some users[95]. Quite simply, a large number of contour levels are visualized at once, with two colors. However, the user still sometimes needs to change contour level, as a final cutoff (the outermost shell) is still needed.

If volumetric ray casting is to be seriously considered for model building, there is a technical hurdle to be cleared too: it requires that the visualization of the model be superimposed on the visualization of the data. This is difficult, because in volumetric ray casting, the color that the user sees when looking at a given point - i.e. along a particular way - is calculated “all in one go” in a very computationally intensive process. If one were to try to take an off-the-shelf ray casting implementation (as I have in figure 63) and try to superimpose it on the model atoms, the atoms would appear either completely in front of, or completely behind, the data.

10.1.3 Slicing

Slicing is the preferred method of scalar field visualization for MRI data. It is not used in model building very commonly, but can certainly be useful sometimes, and so is worth discussing.

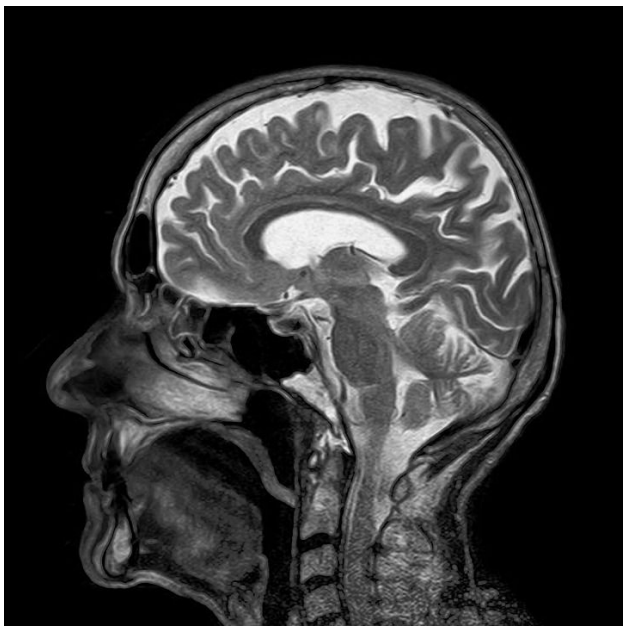


Figure 65*: a slice, directly down the middle, of an MRI scan[96]; the totality of the dataset will have been a scalar field of the same kind as the electron density maps, except that MRI measures water density. The structures seen are very varied but have a specific expected appearance, so abnormalities can be identified quickly.*

I define slicing as taking any 2-dimensional subspace of the 3D scalar field and then coloring every point on the subspace depending on the value of the scalar field at that point. MRI scans such as figure 65 take a flat slice, but curved slices are also allowed, see figure 66.

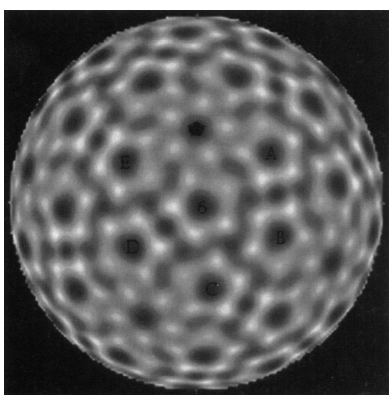


Figure 66: Electron density map containing bovine papillomavirus[97], where the author has decided to display a spherical “slice” of the map (because the virus is spherical); slices do not have to be flat as MRIs usually are, but can be any 2D surface

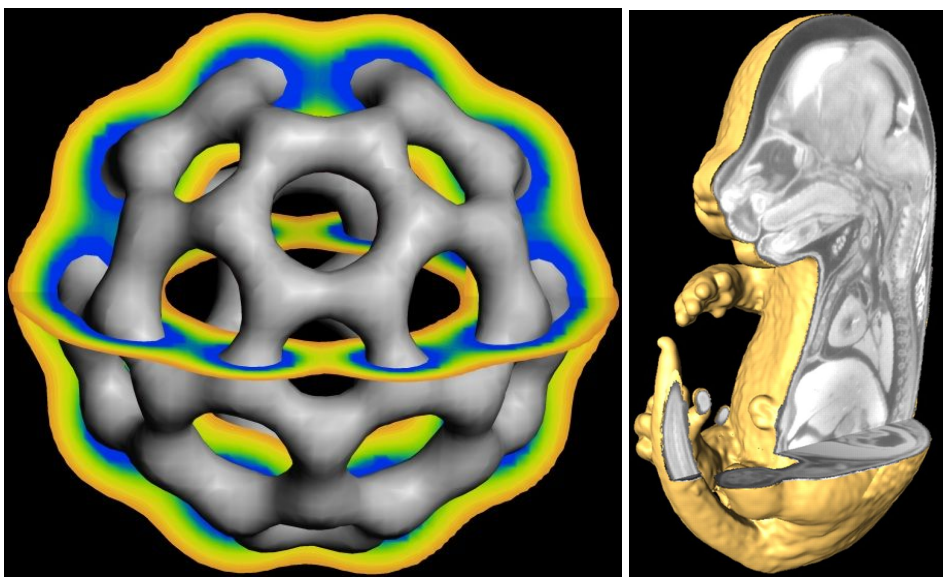


Figure 67: planar slices combined with contouring in an understandable way. Left: a “buckyball” [98] Right: a mouse embryo [99]

One idea that I argue should be explored is similar to figure 67. The idea is that one could add, going along every bond in one’s model, a rectangular “slice”, on which the map values are visualized (either as color or as opacity for some colored fog). Each rectangle would always be oriented towards the camera. This could potentially allow the user to easily get a goodness-of-fit estimation for each bond.

10.2 Visualization of model

While visualizing a model is not as complex as visualizing the density data, it raises a few questions and so is worth discussing.

The “ball and stick” and “licquorice” / “bond-only” representations are what is most often used when a molecule is considered at the level of detail that model refinement involves. The ball-and-stick representation is not realistic; in the real world, the atoms have an impact on the shape of a large “molecular orbital”, which bulges in complex ways that correspond to atoms. In theory, for a given PDB file, this molecular orbital could be calculated and visualized. But such a realistic representation would be harder to think about, whereas ball and stick representations are easy to think about, because spheres and cylinders are in a sense the simplest shapes one can imagine that still communicate atom and bond positions. It is also straightforward to visualize this, as threejs has a built-in “sphere” and “cylinder” function.

“Shading model” has an impact on how easy it is to interpret a 3D image, see figure 68. I use the “lambert” shading model, which is simple, computationally efficient, and gives at least some lighting information. I considered being more ambitious: the rendering in figure 68 involves “ambient occlusion”, where every atom potentially obscures some light from touching every other atom, creating “soft shadows”. Making this work efficiently in our program would have taken a great deal of time (probably requiring a complex “signed distance function”) and might have risked going below the 90fps limit.

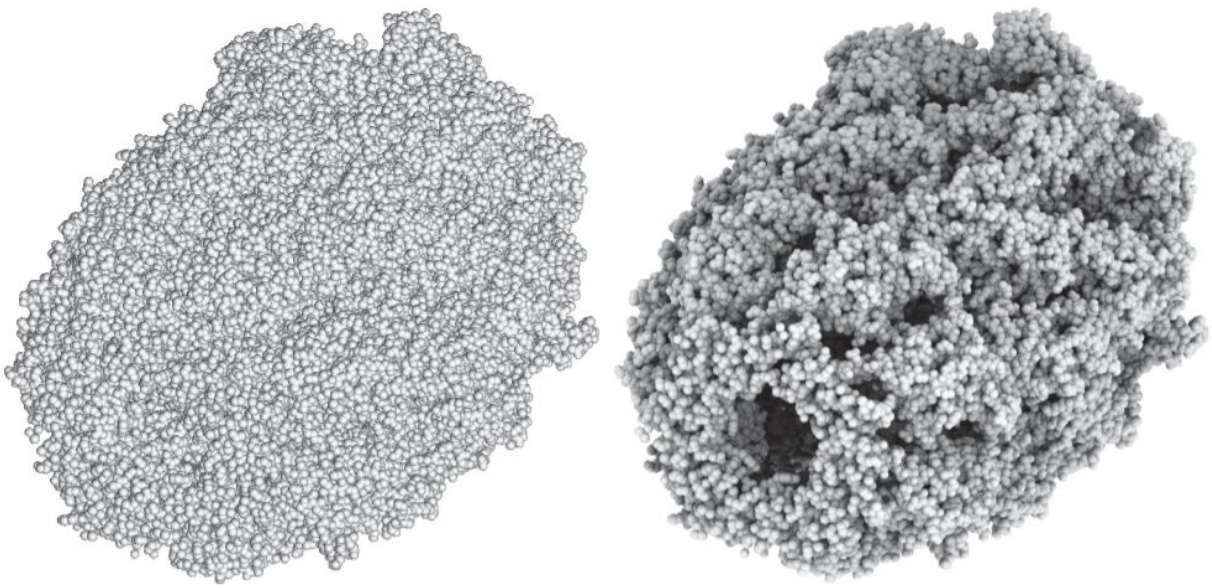


Figure 68: these two pictures actually have the same atoms rendered in the same places. On the right there are extra lighting effects which make it significantly easier to immediately see the general 3D shape[100]. Structure from [101].

One major technical hurdle is the fact that, for the sake of computational efficiency, ideally the list of atoms to be displayed has a fixed length. Published benchmarks[102], and my own, suggest that this is at least 4 times faster than any alternative, keeping the number of atoms equal. This essentially means that the way that atom positions are handled has to keep room for “atoms that do not exist yet”, and when an atom is “deleted”, parts of it are still in memory, it just has to be skipped over when being displayed or saved to a file. This becomes especially difficult because when CootVR is connected to Coot, because it means that there are three separate arrays of atoms to be synchronized: the array of atoms as stored by Coot; the array of atoms as stored by CootVR; and the array of atoms as *displayed* by CootVR (Coot also has a separate “displayed” array, but at least I do not interact with this).

11 Communication with Coot

CootVR can, with an optional script, be connected to Coot. There are various useful functions that this enables, because Coot itself contains many hundreds of features, well-used by structural biologists, that have been developed over the last twenty years.

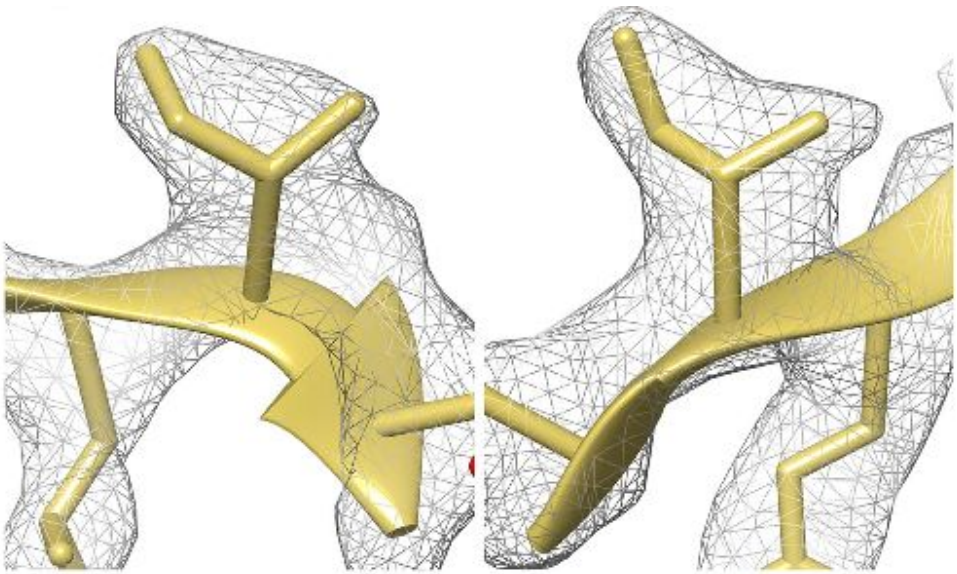


Figure 69: Two different rotamers for a single amino acid sidechain, from [103].

To give an example, Coot has “rotamer libraries”, and an interface for searching them. As described above, a given amino acid side chain is said to have different “rotamers” for its atoms. “Rotamer libraries” contain experimentally-observed low-energy-states for those atom positions. CootVR, if and only if it is connected to Coot, can use this Coot function - the user can pick up a spherical tool, overlap a residue, and use it, which causes it to snap to the correct rotamer. I considered making the tool instead be “cycle rotamer” to go through the library, or to let the user grab the atoms and move them around, and I would fit the closest rotamer in the library - however, “autofitting” is faster.

Since CootVR is intended only as a supplement to Coot, going back-and-forth between modelling in Coot and in CootVR is meant to be an option. Another interesting benefit of Coot-CootVR interaction is that in principle it allows multiple people to use Coot simultaneously, working on the same model. Collaborating with Coot, i.e. having multiple structural biologists at a single screen making suggestions for how a model should be built, and discussing options for it, is a quite common. When this is done, Coot has a significant (and interesting) drawback, which is that the “Coot shake” (described above) that is used to obtain a 3D impression of the scene only appears to work for the user who is holding the mouse and performing the shake. A Coot/CootVR combination works better in this way - both users can be looking at the same

molecule but have their own perspective on it and be building up their own 3D picture of it. Ours is not actually the first program to try to solve this problem in this way - two other Coot add-ons did the same thing[104,105], though not using VR.

Interfacing with Coot creates a number of hurdles. Again, VR currently only works on windows, but the most recent builds of Coot work on Linux, so there is the need to use a Linux virtual machine. Coot does not recognize Javascript, the language that CootVR is mostly written in, so I had to write a python script that acts as a bridge between Coot and CootVR. Finally, it is necessary to maintain synchronization between the states of Coot and CootVR, which is quite involved, because the arrays of atoms are stored in different formats and Coot works at a much lower framerate than CootVR.

11.1 Influence of CootVR on Coot

Interestingly, some changes have been made to Coot as a result of discussions concerning CootVR.

One set of changes has made Coot's structure slightly more like that of a video game. The "main loop" of a video game is simple: input (eg button presses and mouse movements) is read; "objects" in the virtual world are "simulated", possibly changing based on the input; and then the "frame" is "rendered". The goal of the programmer is then to ensure that this happens a minimum of 30 times a second, to maintain the impression that the virtual world is real. The virtual world is given a visualization that is composed to give a certain impression of how it works.

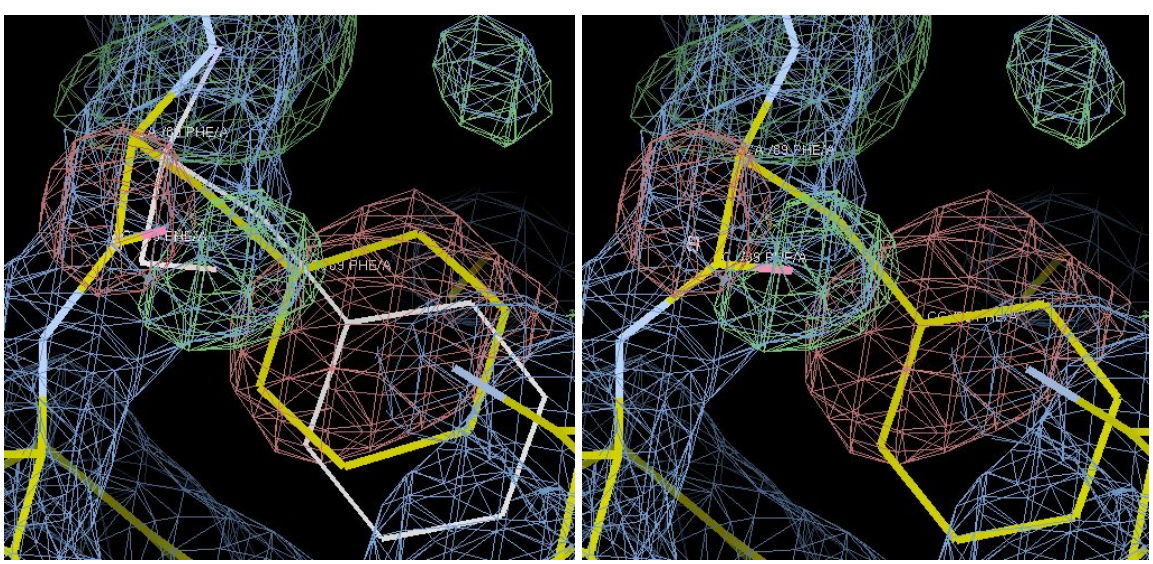


Figure 70: left: the way that Coot used to visualize refinement; right: the current way

One aspect of Coot's visualization in 2016, before the CootVR project began, was that when the user enabled refinement, a second visualization of all of the refining atoms would appear on the screen, in white. The justification for this was not based on visual clarity or helpfulness with the interface, but simply on the fact that when Coot performs refinement, it makes a separate array of atoms, and it seemed logical for the visualization to reflect this. However, this creates a cognitive load and makes the visualization harder to understand. One principle relatively common in the games industry is "The way things appear is not the way that they have to be programmed"[105,106]. After conversations in which this principle was discussed, the refinement visualization was simplified by redrawing the molecule without the atoms being refined.

Again regarding refinement, it was previously the case that Coot would freeze, or at least go down to a significantly lower framerate, when a lot of atoms were refined. The reason for this was because refinement can be computationally expensive, and was only single-threaded, the refinement being in the same thread as the OpenGL updates. Any update to the view would need the refinement to be stopped for the bonds to be recalculated and drawn, and then restarted afterwards, which would take 10ms. Thus the user had to choose between having the refinement be either "smooth" or "fast".

I suggested that Coot should adopt a more games-industry-oriented view of visualization programming. By this I mean stating a 30 frame per second ideal, and having as many atoms refine as possible, at all times, seeing their movement. The Coot infrastructure was reworked to put refinement in a different thread to the bond recalculation and OpenGL drawing thread. The refinement rewrite allowed a change to the way the atoms updated: instead of pulling on an atom and then releasing (similar to the gam Angry Birds, pulling on the rubber band of the sling shot and then letting go) the atoms are now refined on every movement of the mouse whilst dragging an atom(similar to dribbling in a football game, with a ball constantly responding to the position of a player's foot).

Refinement is not a component of the Coot-CootVR connection, but if it was, our idea was that CootVR would work similarly, with the set of atoms near the user's hand always refining.

I also suggested other game-like enhancements to Coot's interface such as atoms in energetically-unfavourable states vibrating or even having frowning faces on them. This is only half-serious, but is plausibly a good idea because searching through coot menus is in a sense "looking for problems", when really the interface should take it upon itself to prioritize and highlight them[63]. This is arguably what motivated the invention of "Ramaballs" (coloring carbon-alphas by amino acid ramachandran score) in Isolde, which have since been adopted by Coot.

12 Assessment

In a manner of speaking, our project is exactly one data point in a larger experiment. In the broadest terms, the question of interest is “is VR more useful for scientists than the mouse/keyboard interface?”. In less broad terms, the question is “is VR more useful for structural biologists than the mouse/keyboard interface?”, and in narrow terms it is “is VR more useful for biomolecular model building than the mouse/keyboard interface?”. But this narrowing-down has been done based on the fact that, as I see it, *if* the answer to the first question were to be “yes” for any particular task in any particular field, I would expect it to be biomolecular model building, since it is a pathologically 3D-based, pathologically manual task, a domain where VR can excel.

I assessed the value of the software in several ways.

12.1 Community reception

CootVR has been shown to a large number of structural biologists across four conferences and three laboratories. Responses are polarised, though mainly positive. Some users are so enthusiastic about the idea that their sense of fun overwhelms their practical thinking, and they argue for the implementation of a tutorial involving a cartoon character. At the other end are people who are very cynical, refusing to believe that VR could be even remotely useful.

I have found that part of the reason for extreme cynicism is the impression left by a few of the experiments with structural biology and VR such as those described in the literature review, because of their vague claims that biologists will “understand” their molecules better once they are physically inside them. In some cases, once the person was assured that there were concrete claims (such as that the protein painter would be faster than what Coot does in some situations), their view softened.

Age appears to be a factor in whether a person expects benefits from VR, probably partly due to a misguided desire on the part of younger people to play with recent electronic “toys” and partly because of an unwillingness to change on the part of older people. In the early 2000s, Coot itself was a new program and had “win out” over its competitor, O; at that time, it was younger members of the community who were early adopters. The vast majority of biologists would now agree that Coot’s superiority should have been noticed sooner (it always had a superior lighting system, and therefore superior 3D readability than O). So on that occasion people who would “stick in the mud” proved to be wrong. With this said, I believe that a certain amount of curmudgeonliness is appropriate, to the extent that the opposition forces people with new ideas to clarify them and think up tests that could prove a curmugeon wrong - that is what I have endeavoured to do above.

Figure 71: CootVR at a conference

I decided to conduct a questionnaire regarding CootVR; figure 72 shows the major result. Each questionnaire participant was shown CootVR at an early stage, and did a bare minimum of interaction with it, in all cases at least using the rigid mover tool. The response is extremely positive. There is likely to be some distortion of the results and so it should not be considered a representative survey of the structural biology community; the group came from among Coot-using structural biologists at conferences, but they chose to approach the CootVR “stall”, and were therefore self-selecting as being at least somewhat intrigued by the prospect of using CootVR. Additionally, since they were speaking directly to its creator and were grateful for having used it, they may have felt social pressure to be polite and say that they would use it.

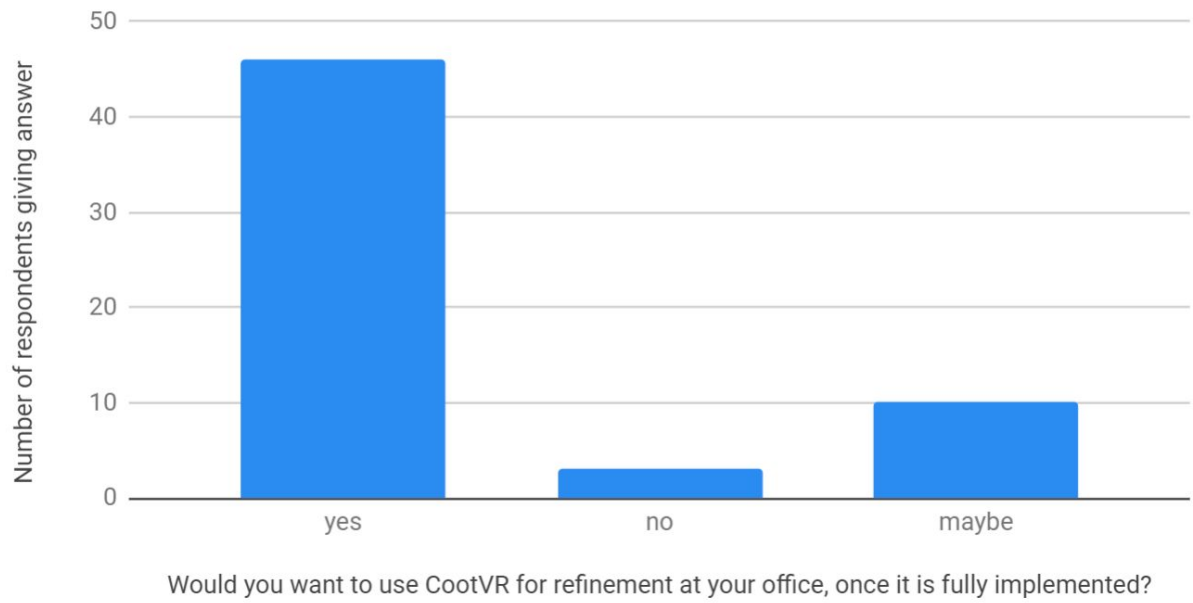


Figure 72: Survey results

Many people who are experienced with other molecular graphics programs have remarked on how CootVR does not feel cluttered, even though I made no particular effort to avoid clutter. People tend to look at more atoms at a time with CootVR than Coot, which is better for making decisions about the positions of multiple atoms at the same time.

12.2 Case study

I took it upon ourselves to build, from the ground up, a structure using CootVR, in order to get a subjective sense of what it is like to use it for a protracted length of time (and to give us ideas for improving our tools).

At first I was going to use high-resolution crystallographic data, and some was procured. I decided that this was a bad idea - CootVR, it should be admitted, is not currently suited to the kinds of small adjustment that are needed in this kind of data, which is mostly automated. I instead switched to an electron microscopy dataset[107], one for which there is no agreed-upon model that could be “cheated” with.

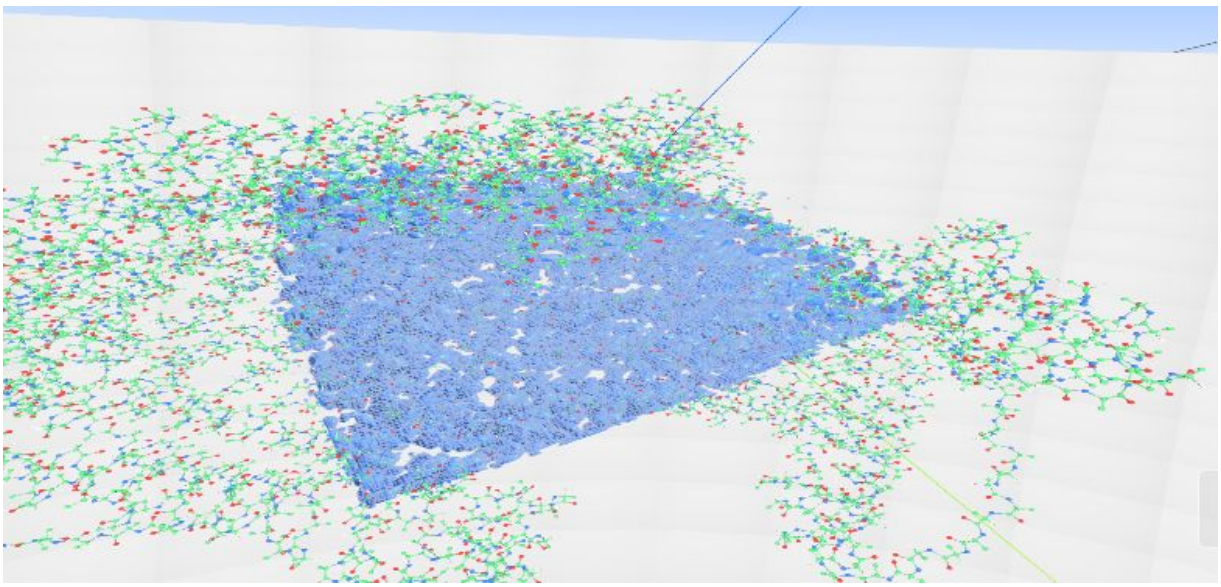


Figure 73: Our case study protein in a state of partial completion.

The main tool used in the case study was the protein painter, described above. The protein complex in the dataset, human apoferritin, contains 24 copies of the same protein. They are related by symmetry, but since this is an exercise in seeing how much can be done, I decided to create all of them by hand.

It should be noted that the model that I eventually created is a very gross approximation to the protein itself. If I were seriously considering this as a model, I would run it through various refinement programs to ensure that it keeps within all known chemical constraints (which it currently does not).

The main interesting result was how quickly I found I was able to make decisions. At our fastest, I estimate that I was able to place one amino acid every two or three seconds; again I do not believe that the decisions were necessarily very good, but they were enough to go along with.

It had previously been suggested that alpha helices (which apoferritin has many of) would be difficult to deal with in the context of CootVR - actually it turned out that they were extremely easy, more easy than non-helix areas of protein. The reason it was believed that they would be difficult was because alpha helices, in the context of non-VR-Coot, have been reported to be difficult to build by adding proteins one-by-one. I suspect that this is due to the fact that alpha helices are innately 3D objects. In the context of CootVR, the reason that they are so easy to build is because they are very repetitive; there is one specific motion with the hand to rotate model as desired, and then another to place the amino acid.

Interestingly, with many amino acid placements, it is barely even necessary to do a second hand movement - one can simply hold one's "painting hand" in place and rotate the model such that it is in the correct position relative to the rest of the model. The thing that ends up

happenning “naturally” is that the molecule-moving hand would take care of large movements while the painting hand took care of minor movements. This was effortless to get used to.

For comfort, I found:

1. It was good to have the model be scaled such that an Ångstrom would be 12cm in relation to our hands.
2. It was necessary to rest our elbows on a hard surface; when I did not do this, our wrists and upper-arm rapidly became fatigued.
3. It was good to keep interactions to bouts of approximately 15 minutes. Any more and I found that there was a small amount of eye strain. This may have been due to dryness inside of the headset, or possibly because I was not blinking as much as usual.
4. It was good to keep the model very very close to our face, within a few centimeters, probably because (unconsciously) I wanted to get as much information as possible from stereoscopy (this may have contributed lead to the aforementioned eye strain). It is interesting to note that if this is a requirement for VR-based model building, then light field displays (described above) may not be a very good replacement for headsets.

12.3 Timing

For reasons alluded to previously, I believe that the major value proposition of VR-based software is the possibility that it might *speed up* certain tasks. Speaking to one Coot user who worked for the pharmaceutical company Bosch, he said he was skeptical that he would use CootVR - but when asked “would you use it if it could be demonstrated to be 20% faster than Coot?” replied “in that case, yes”.

CootVR cannot be used in a test of VR against mouse/keyboard that can be fully persuasive, because it does not have all of the capabilities of Coot. For example, it does not have the ability to check molprobity score, or the ability to label residues (a task that requires typing on a keyboard). For these tasks, by definition CootVR is inferior to Coot, and so CootVR, at least in its current form, cannot fully be considered a “VR version of Coot”.

Even if it could, it would take a large amount of time as a whole; to comprehensively answer the question, ideally one would have several controls:

1. How much practice a person has had with each program (a person who is already proficient enough with Coot, using keyboard shortcuts and so on, and is towards the end of their career, may have nothing to gain from switching to CootVR)
2. How deep the person’s understanding of structural biology in general is, or their level of geometric intuition.
3. Whether a user has a physical disability that stops them from using two hands simultaneously
4. Number of atoms in the biomolecule
5. The resolution of the dataset
6. Whether the dataset contains problem cases, such as carbohydrate ligands

7. How hard the operator is concentrating - people may be faster or slower at using Coot at different times of day, and people might be faster or slower at using CootVR depending on whether they are excited by it (the “novelty effect”).

There is a narrower but still interesting and impactful hypothesis that can be tested using CootVR in its current form though: “is there any common biomolecular modelling task for which VR is better than a mouse-and-keyboard interface?”. I have conducted an experiment that I believe answers this question.

The biomolecular modelling task I have focussed on is creating new chain ab-initio, with the chain not being connected. This is a very easy experiment to set up; I simply needed to take some unseen datasets with model and map, and to remove some amino acids from the chain. This was done, and timings were taken for people filing in the chain using CootVR and Coot.

It was decided that only “proficient” users of Coot and CootVR would be suitable. In the case of both programs, they are used by professionals who are comfortable spending several months becoming proficient, and so nobody would be persuaded by an experiment that showed that either CootVR was better for amateur use. There is only one person who is a “proficient” user of CootVR, which is the present author; this is unfortunate but unavoidable. I did a lot of “practice” before doing the experiment, including the case study above.

I considered comparing “fit” to final data, but decided not to, as the goal of drawing the new chain at this stage was not to focus on details but give a general shape for the chain, and then refine it in detail using other tools afterwards.

The following table encapsulates our results:

Puzzle	Number of residues	Coot result (seconds)	CootVR result (seconds)
1	4	36	30
2	8	180	25
3	25	1140	72

On both the Coot and CootVR sides, mistakes were made and some “undoing” was needed - this is normal and is included in the timings. In all three cases, on the CootVR side, the task was brief enough that no break was needed, which would require undonning and re-donning the headset, which would have taken up a large amount of time.

I note that CootVR was quite a lot faster than Coot on average, and so conclude that at least for this usage VR is superior to the mouse/keyboard interface.

13 Conclusion

13.1 “Realism”/”skeumorphism” as an interface design goal

In discourse around user interface design for VR, people frequently advocate that interacting with the interface should feel quite similar to interacting with real-world objects[108]; this was a design goal that I was, for almost all of the development of the program, at least slightly influenced by. “Skeumorphic” is a word in interface design meaning “modelling the real world”, and was advocated at least to some extent by Steve Jobs[109–111].

To give an extreme example of a principle like this, in two presentations at the same year at the Virtual Reality Developer Conference, it was argued that “floating text not associated with a normal physical object” should be avoided[109,112]. Suppose that I wanted to give a program the functionality “close current session”. A simple and obvious way to do this, and the way that I would do it with the panel described above, would be to make a sign saying “close current session” and have it be the case that if the user pointed at this sign and pressed a button it would have this effect. It would be argued that this is bad though. Instead the two VR programs that were talked about on this occasion had an object called an “exit burrito”, see figure 74.

Figure 74: The “exit burrito” in the VR program “Job Simulator”. The burrito would be within reach of the user, who would physically pick it up and raise it to their mouth, “eating” part of it; this would reveal the “bitten” texture saying “really?” spelled out in rice; if eaten again, the user would leave the current session.

The fundamental reason that this solution was advocated was that it does something to model real-world gestures and events such as eating and blacking out from eating something. This would be argued to make it more intuitive, natural, and comfortable; in real life, one does not

just point at signs and will them to do something and expect a result. It is an extreme example because, to some extent, it must have been thought of as a practical joke that was guaranteed to feel original (the two programs it was in were some of the first to appear on the VR market). But it should be understood that this was also, to a large extent, made as a serious proposition for future UI design[109]. I would say that this has influenced other VR structural biology programs; Molecular Rift, for example, has various functions executed through hand gestures which could have been simple buttons[38,109–111].

I considered, but did not implement, “eating” as an interaction method, but there were several things I did which were comparable:

1. In designing the protein painter, as described above, I wanted to have no button input and have all input be “gestural”, as if the object was non-mechanical and interacted with purely through movement, like a rope.
2. Originally, instead of having a panel, I was going to have a flat desk surface and have all of the hand tools be physical objects, complete with a gravity simulation, that would make them fall onto the desk when let go. Interestingly this idea has been around before VR in the form of the experimental “bumptop” interface[109,110].
3. One of the early tools I implemented was a “laser pointer”, simply intended for the user in VR to point at different atoms as a spectator sees them. On this tool I had a button which would be pressed to make the laser appear; but really there was no need for it to be there, as there would be no situation in which you would be holding the laser pointer but not want it to be “switched on”.
4. The “visibox”, described above, originally being a kind of lantern.
5. Arguments made in favour of visualizing the electron density map as a gaseous volume
6. I spent a large amount of time trying to think of an appropriate “analogy” for tool use, prototyping an idea where you would always be holding an electric-drill-like object whose “drill bit would be changed” when you picked up a new tool; I even considered having “miming drill bit changing” be the way that tools were changed.
7. Relatedly, I had an “environment distances” tool that was meant to behave somewhat like a “staple gun”; the user would be able to make “clone” the tool and attach them to any amino acid for which they wanted to see environment distances; these clones would remain attached to the amino acids until removed.
8. The way that the user changes the isolevel in the current version is straightforward: you push the analogue stick up to increase it and down to decrease it. But I previously considered having the user reach out to grab the isosurface and pull it in the direction they wanted it to go, in mimicry of grabbing a sheet of rubber.
9. In the real world, when altering the shape of something small (perhaps when making origami or woodwork), one usually has to hold it with both hands, because one hand needs to stabilize it while the other does a fine motion to it that needs force applied from a certain direction. This is not necessary in VR - the work of the “stabilizing hand” can be done automatically. This affected how I designed the protein painter.

To reiterate, I believe that the most important goal of a user interface is to help the user do what they need to do quickly; VR as a tool is only valuable to the extent that it is, at least for some tasks, better according to this metric than a keyboard-and-mouse or touchscreen setup. It may be argued that there are some situations in which skeumorphism does improve efficiency, but in our experience it has mainly worked against it. It would appear that the most efficient way for a user interface to work will in general involve highly “unrealistic” interactions. I would say that even the goal of being “intuitive” can work against efficiency - for example, in spite of having moved and rotated the molecule with one hand, many users would often have the “realistic intuition” that they needed both hands to move the molecule. I could have “pandered” to this by making it so that both hands were necessary - but they are not, and it is a good thing for people to get used to using only one hand.

Another way in which realism or “immersion” might be a negative is that it may be the case that the more “real” an object feels to a user, the stranger it is that they have the ability to linearly scale it. Linear scaling never happens in the real world, but is a constantly-used action in CootVR.

There were some aspects of CootVR that made it more realistic than Coot, like the fact that there is a perspective projection on the camera instead of an orthographic one, and detailed lighting on the molecule and map. These are purely visualization touches though, not user interface ones.

13.1.1 Software as environment

In literature looking at the psychology of VR software, and in some papers on VR and structural biology, much is made of the fact that, in VR, the software tool is synonymous with the environment in which the user finds themselves.[113] This is the most significant example of the desire for realistic analogies in VR UI design.

Figure 75: a kitchen and electronics workshop, both used as analogies for things that software development should move towards [62]

The proposed advantages of the “software as environment” paradigm include:

1. By making the user move around more, they will get exercise, which is healthy
2. Increased movement may also cause the users to enjoy themselves more
3. Users may have a better memory for where things are, which allows them to work faster
4. Users may find the software easier to learn, because 3D environments take advantage of things that the body naturally does and inferences it naturally makes

There are at least three major decisions in the design of our software, described above, where I have specifically gone *against* this thinking:

1. I center the panel on a particular location that I expect the user's head to sit
2. I use clipping planes to limit what the user can see, instead of having the molecule take up all of the space around the user. Additionally the clipping planes are "stuck in place" (their shape and size can be changed but not their position)
3. I make it so the user can effortlessly rotate and scale the molecule in order to look at different parts of it, instead of them moving their own head or body over to that part.

The purpose of a user interface is to allow the user to do what they need to do as efficiently as possible, and for reasons described in their respective chapters, I found all of these to be clearly the best decision in context. It has been argued that the desk-based office design is what is holding development in this direction back[113]. But our intuition is that even if I had complete freedom with a space to build anything in it with current VR technology, for this application I would still build a desk.

13.2 Dimensionality-reducing UI design

There are a number of decisions and redesigns with CootVR's interface which I noticed a pattern running through. The below are all examples:

1. Having the various buttons appear on a panel around the player rather than on their wrist
2. Where possible, putting things on the 2D panel rather than having them hover in 3D
3. Removing the "lantern" idea from the visibox, i.e. making it so that the visibox stayed in place and could not be moved by the user independently of the molecule
4. Also on the visibox, making it so that, when it was resized, the resizing keeps the visibox mirror-symmetrical
5. In implementing the "protein painter", deciding to use only hand position as a control, rather than position and orientation.
6. When using "rotamer libraries", it could have been the case that the user has to drag atoms around and the "nearest" rotamer is selected; but it is better to simply have a single button to cycle them.

The thing that all of these have in common is "dimensionality reduction"; I claim that this is a non-trivial principle that is worth adopting widely, especially for VR developers. In all cases, the user is giving input to the program, and there is a choice of what kind of input they should give. I would say that, other things being equal, the ideal input is the one that requires the user

specifying a point in the lowest-dimensional space. To give the most basic example, if a piece of UI may require giving either a single numerical value or a pair, it is better to only have them specify a single numerical value.

Figure 76: when middle-clicking on a webpage in windows 10, the mouse cursor will turn into this icon, allowing the user to scroll the webpage with mouse input. I would say that, when this happens and the page cannot be scrolled horizontally, it would be better if the icon could not be moved horizontally, because this involves the player giving more input than is needed

This matters a great deal in the context of VR because a VR controller increases the number of dimensions that the user's input has over a mouse; our earlier statement that "VR improves 3D articulation" is essentially identical to this statement. So when designing an interface and considering whether to use a VR input method or just the mouse, the question becomes "is adding an extra dimension to the input necessary, given the extra time investment from the user to specify its value?". The answer to this question might be complicated. For example, a UI designer may have a choice between either having two input dimensions or having a single dimension, and then having the second input dimension be automatically deduced when the user presses a button. I would say that in this situation, the latter option is likely to be better, even if it would appear that the first option "feels better".

This principle has an important implication for the place of VR in structural biology. Specifically, when I began the project I believed that selecting atoms and residues in a molecule would be better in VR than with a mouse and a 2D screen. I now believe that it is actually worse, because it involves the user specifying more information than is necessary - the mouse is perfect for pointing at an atom within one's field of vision, which is fundamentally 2D, because the program can simply look along the "ray" that is cast by the mouse in 3D space. In VR, the user must put their hand at the correct position in 3D, which takes a little more time[114].

13.3 Closing remarks

I should strain that I do not, at this time, actually recommend using CootVR for model building. There are two technologies that could change this, which are light-field displays and augmented reality glasses. These would fully remove the donning problem and make it so that “VR” is seamlessly integrated into ordinary user interfaces. When this happens, I recommend that it be integrated into Coot as soon as possible and that all structural biologists purchase the technology, and that a protein painter tool be implemented within it following our design.

It was our opinion during the project that the most interesting new thing to explore was the hand control. I would say that in actual fact it has fairly limited usage, fundamentally because of the dimensionality-reducing principle that suggests that a mouse is better for choosing among options and atoms.

The hand controllers are not the only interesting things about VR by any means though; head control sensetivity and perfect stereoscopy are very important too. They enable 3D graphs (which I did not implement), but more importantly they allow for a great deal more visual “decoration” of the molecule. For example, currently in Coot, “environment distances” is enabled on a per-residue basis. I would say that with the benefits of VR, environment distances can be enabled on the whole molecule.

To speak more broadly, I have seen requests for VR interfaces in some situations where it is definitely not appropriate, including “conducting a search on the PDB for donut-shaped proteins” and “wanting to estimate how many rubisco complexes could fit into a bionanocontainer”. Instead of expecting it to do anything new, hand input in particular should be considered a “last resort” for those tasks which cannot be automated and for which the mouse is very cumbersome to use. I would recommend that VR developers avoid being overambitious about what VR can achieve within science, and to focus more than anything else on how it can save time. Some developers I speak to have the impression that VR can enable things that are completely impossible before its invention; I would say that this expectation should be completely disregarded. Being “easy to learn” is not remotely a selling point for serious scientific software, and this rules out much of the appeal of VR.

Fundamentally VR is only a visualization and input method; at the inevitable point where light-field and AR displays become ubiquitous, very little will change for scientific software. Even in the pathologically-3D domain of model refinement, which unlike the enormous majority of scientific computing involves manual 3D manipulation, opportunities for benefits are rare, although it is interesting to see that they are not unheard of.

14 Bibliography

1. Tame, J. R. H. & Vallone, B. The structures of deoxy human haemoglobin and the mutant Hb Tyrα42His at 120 K. *Acta Crystallographica Section D Biological Crystallography* vol. 56 805–811 (2000).
2. Krone, M., Grottel, S., Reina, G., Muller, C. & Ertl, T. 10 Years of MegaMol: The Pain and Gain of Creating Your Own Visualization Framework. *IEEE Comput. Graph. Appl.* **38**, 109–114 (2018).
3. Moritz, E. & Meyer, J. Interactive 3D protein structure visualization using virtual reality. in *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering* doi:10.1109/bibe.2004.1317384.
4. Wriggers, W. & Birmanns, S. Using situs for flexible and rigid-body fitting of multiresolution single-molecule data. *J. Struct. Biol.* **133**, 193–202 (2001).
5. Richardson, J. S. & Richardson, D. C. Doing molecular biophysics: finding, naming, and picturing signal within complexity. *Annu. Rev. Biophys.* **42**, 1–28 (2013).
6. Surles, M. C., Richardson, J. S., Richardson, D. C. & Brooks, F. P. Sculpting proteins interactively: Continual energy minimization embedded in a graphical modeling system. *Protein Sci.* **3**, 198–210 (2008).
7. Iakovou, G., Hayward, S. & Laycock, S. D. Virtual Environment for Studying the Docking Interactions of Rigid Biomolecules with Haptics. *J. Chem. Inf. Model.* **57**, 1142–1152 (2017).
8. Land, H. & Humble, M. S. YASARA: A Tool to Obtain Structural Guidance in Biocatalytic Investigations. *Methods Mol. Biol.* **1685**, 43–67 (2018).
9. Kingsley, L. J. *et al.* Development of a virtual reality platform for effective communication of

structural data in drug discovery. *J. Mol. Graph. Model.* **89**, 234–241 (2019).

10. *VX nerve agent - Periodic Table of Videos*. (2017).
11. Chimera Movie Making. https://www.cgl.ucsf.edu/chimera/data/nih-oct2012/movies.html.
12. Chimera Movie Making. https://www.cgl.ucsf.edu/chimera/data/nih-oct2012/movies.html.
13. Oculus Rift Molecular Visualization. http://www.cgl.ucsf.edu/chimera/data/oculus-jan2014/oculus.html.
14. O'Connor, M., Tew, P., Sage, B., McIntosh-Smith, S. & Glowacki, D. R. Nano Simbox. in *Proceedings of the 3rd International Workshop on OpenCL - IWOCL '15* (2015). doi:10.1145/2791321.2791341.
15. Persson, P. B. *et al.* Designing and Evaluating a Haptic System for Biomolecular Education. in *2007 IEEE Virtual Reality Conference* (2007). doi:10.1109/vr.2007.352478.
16. Website. Use of a Three-Dimensional Virtual Environment to Teach Drug-Receptor Interactions Read More: https://www.ajpe.org/doi/full/10.5688/ajpe77111.
17. ChimeraX Cellphone VR Movies. https://www.rbvi.ucsf.edu/chimerax/docs/user/vrphone.html.
18. Protein Viewer. http://hamishtodd1.github.io/HepCardboard.
19. Sable, R. & Jois, S. Surfing the Protein-Protein Interaction Surface Using Docking Methods: Application to the Design of PPI Inhibitors. *Molecules* **20**, 11569–11603 (2015).
20. Dominguez, C., Boelens, R. & Alexandre M J. HADDOCK: A Protein−Protein Docking Approach Based on Biochemical or Biophysical Information. *Journal of the American Chemical Society* vol. 125 1731–1737 (2003).
21. *Bioblox VR - A Protein Docking Game (Full Demo)*. (2017).
22. Anderson, A. & Weng, Z. VRDD: applying virtual reality visualization to protein docking and design. *J. Mol. Graph. Model.* **17**, 180–6, 217 (1999).

23. Férey, N. *et al.* Multisensory VR interaction for protein-docking in the CoRSAIRe project. *Virtual Reality* vol. 13 273–293 (2009).
24. Cakici, S., Sumengen, S., Sezerman, U. & Balcisoy, S. DockPro. in *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry - VRCAI '08* (2008). doi:10.1145/1477862.1477877.
25. Levine, D. *et al.* Stalk: an interactive system for virtual molecular docking. *IEEE Computational Science and Engineering* **4**, 55–65 (1997).
26. Subasi, E. & Basdogan, C. A New Haptic Interaction and Visualization Approach for Rigid Molecular Docking in Virtual Environments. *Presence: Teleoperators and Virtual Environments* **17**, 73–90 (2008).
27. Nagata, H., Mizushima, H. & Tanaka, H. Concept and prototype of protein-ligand docking simulator with force feedback technology. *Bioinformatics* **18**, 140–146 (2002).
28. Olmez, E. O. & Akbulut, B. S. Binding Protein. in *Binding Protein* (IntechOpen, 2012).
29. Zumla, A., Chan, J. F. W., Azhar, E. I., Hui, D. S. C. & Yuen, K.-Y. Coronaviruses - drug discovery and therapeutic options. *Nat. Rev. Drug Discov.* **15**, 327–347 (2016).
30. Yu, M. J. Druggable chemical space and enumerative combinatorics. *J. Cheminform.* **5**, 19 (2013).
31. Chen, V. B. *et al.* MolProbity: all-atom structure validation for macromolecular crystallography. *Acta Crystallogr. D Biol. Crystallogr.* **66**, 12–21 (2010).
32. Emsley, P., Lohkamp, B., Scott, W. G. & Cowtan, K. Features and development of Coot. *Acta Crystallogr. D Biol. Crystallogr.* **66**, 486–501 (2010).
33. Jones, T. A., Alwyn Jones, T., Bergdoll, M. & Kjeldgaard, M. O: A Macromolecule Modeling Environment. *Crystallographic and Modeling Methods in Molecular Design* 189–199 (1990) doi:10.1007/978-1-4612-3374-9_13.

34. Croll, T. I. ISOLDE: a physically realistic environment for model building into low-resolution electron-density maps. *Acta Crystallogr D Struct Biol* **74**, 519–530 (2018).
35. Goddard, T. D. *et al.* Molecular Visualization on the Holodeck. *J. Mol. Biol.* **430**, 3982–3996 (2018).
36. Kleffner, R. *et al.* Foldit Standalone: a video game-derived protein structure manipulation interface using Rosetta. *Bioinformatics* **33**, 2765–2767 (2017).
37. [No title]. https://www2.mrc-lmb.cam.ac.uk/personal/pemsley/coot/web/coot-nottingham-jan-2010.pdf.
38. Norrby, M., Grebner, C., Eriksson, J. & Boström, J. Molecular Rift: Virtual Reality for Drug Designers. *J. Chem. Inf. Model.* **55**, 2475–2484 (2015).
39. Kay, A. C. Computers, Networks and Education. *Scientific American* vol. 265 138–148 (1991).
40. [No title]. http://worrydream.com/refs/Sutherland%20-%20The%20Ultimate%20Display.pdf.
41. Lanier, J. *Dawn of the New Everything: Encounters with Reality and Virtual Reality*. (Henry Holt and Company, 2017).
42. [No title]. http://worrydream.com/cdg/ResearchAgenda-v0.19-poster.pdf.
43. Frame Rate - an overview | ScienceDirect Topics. https://www.sciencedirect.com/topics/engineering/frame-rate.
44. *Simulation of millisecond protein folding: NTL9 (from Folding@home)*. (2010).
45. Silva, D.-A. *et al.* Millisecond dynamics of RNA polymerase II translocation at atomic resolution. *Proc. Natl. Acad. Sci. U. S. A.* **111**, 7665–7670 (2014).
46. *BioBlox games*. (2017).
47. Skauli. Protein Visualization and Virtual Reality. *lizardphunk* https://lizardphunk.wordpress.com/2015/07/23/protein-visualization-and-virtual-reality/

(2015).

48. *How Augmented Reality Will Change Education Completely | Florian Radke | TEDxGateway*. (2017).
49. Khatib, F. *et al.* Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nat. Struct. Mol. Biol.* **18**, 1175–1177 (2011).
50. Watson, J. D. MOLECULAR STRUCTURE OF NUCLEIC ACIDS. *JAMA* vol. 269 1966 (1993).
51. [No title]. https://www2.mrc-lmb.cam.ac.uk/Personal/pemsley/coot/files/coot-may-2017-ligands-madrid-mcs2017-csic.pdf.
52. Raymond Wagner Phd, Steve Reyntjens PhD Materials & Structural Analysis Thermo. Nobel Prize in Chemistry 2017 Recognizes Key Developments in Cryo-Electron Microscopy. *Research & Development* https://www.rdmag.com/article/2017/12/nobel-prize-chemistry-2017-recognizes-key-developments-cryo-electron-microscopy (2017).
53. | LTS2. https://lts2.epfl.ch/projects/91.
54. Oliva, M. V. & Muhammed, H. H. New Approach for Limited-Angle Problems in Electron Microscope Based on Compressed Sensing. *Engineering* vol. 05 575–578 (2013).
55. Kühlbrandt, W. The Resolution Revolution. *Science* **343**, 1443–1444 (2014).
56. Winn, M. D., Murshudov, G. N. & Papiz, M. Z. Macromolecular TLS refinement in REFMAC at moderate resolutions. *Methods Enzymol.* **374**, 300–321 (2003).
57. Sikosek, T., Miura, S., Tate, S. & Kumar, S. Figure 1.1: The 20 different amino acids occuring in proteins. The. *ResearchGate* https://www.researchgate.net/figure/The-20-different-amino-acids-occuring-in-proteins-The-i

mage-shows-a-protein-chain-in_fig1_264933592 (2012).

58. RNA Chain. http://www1.biologie.uni-hamburg.de/b-online/library/bio201/rnachain.html.
59. Zhou, A. Q., O'Hern, C. S. & Regan, L. Revisiting the Ramachandran plot from a new angle. *Protein Sci.* **20**, 1166 (2011).
60. *Oculus Connect 5 | Medium and Quill for Prototyping with Live Demo*. (2018).
61. [No title]. https://patrickcollison.com/static/files/labs/context.pdf.
62. Seeing Spaces. http://worrydream.com/SeeingSpaces/.
63. Magic Ink: Information Software and the Graphical Interface. http://worrydream.com/MagicInk/.
64. [No title]. http://worrydream.com/TheHumaneRepresentationOfThought/.
65. Perlin, K. Future Reality: How Emerging Technologies Will Change Language Itself. *IEEE Comput. Graph. Appl.* **36**, 84–89 (2016).
66. Sutherland, I. E. A head-mounted three dimensional display. in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)* (1968). doi:10.1145/1476589.1476686.
67. *Oculus Connect 3 Opening Keynote: Michael Abrash*. (2016).
68. Kramida, G. Resolving the Vergence-Accommodation Conflict in Head-Mounted Displays. *IEEE Trans. Vis. Comput. Graph.* **22**, 1912–1931 (2016).
69. [No title]. https://www.gdcvault.com/play/1024533/Lessons-Learned-from-a-Thousand.
70. Shacked, R. & Lischinski, D. Automatic Lighting Design using a Perceptual Quality Metric. *Computer Graphics Forum* vol. 20 215–227 (2001).
71. The Body Is Back! Understanding Embodiment in VR & AR Game Design. https://www.gdcvault.com/play/1023936/The-Body-Is-Back-Understanding.
72. *Oculus Connect 5 | Keynote Day 01*. (2018).

73. [No title]. http://www.nime.org/proceedings/2017/nime2017_paper0008.pdf.

74. *Head Tracking for Desktop VR Displays using the WiiRemote*. (2007).

75. A user study comparing head-mounted and stationary displays - IEEE Conference Publication. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=378265 .

76. Tesler, L. A personal history of modeless text editing and cut/copy-paste. *interactions* vol. 19 70 (2012).

77. Ramachandran Plot. http://www.cryst.bbk.ac.uk/PPS95/course/3_geometry/rama.html.

78. Ramachandran, G. N., Ramakrishnan, C. & Sasisekharan, V. Stereochemistry of polypeptide chain configurations. *J. Mol. Biol.* **7**, 95–99 (1963).

79. Hingerty, B., Brown, R. S. & Jack, A. Further refinement of the structure of yeast tRNAPhe. *J. Mol. Biol.* **124**, 523–534 (1978).

80. Wang, X. *et al.* RNABC: Forward Kinematics to Reduce All-Atom Steric Clashes in RNA Backbone. *J. Math. Biol.* **56**, 253 (2008).

81. Lu, X.-J. & x3dna.org. 3DNA Homepage -- Nucleic Acid Structures. http://x3dna.org/highlights/sugar-pucker-correlates-with-phosphorus-base-distance.

82. pubmeddev & Davis IW, E. al. The backrub motion: how protein backbone shrugs when a sidechain dances. - PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed/16472746.

83. Reynolds, B. E. & Fenton, W. E. *College Geometry: Using the Geometer's Sketchpad, 1st Edition: Using the Geometer's Sketchpad*. (Wiley Global Education, 2011).

84. Blais, B. R. Visual ergonomics of the office workplace. *Chemical Health and Safety* vol. 6 31–38 (1999).

85. Z, I. Artist Desk | Art studio & workspace | Drawing desk, Artist workspace, Workshop studio. *Pinterest* https://www.pinterest.com/pin/450782243924382261/ (2014).

86. van Dam, A. Post-WIMP user interfaces. *Communications of the ACM* vol. 40 63–67

(1997).

87. *Oculus Connect 2: Being There: Designing Standing VR Experiences with Tracked Controllers*. (2015).
88. John Carmack. https://www.facebook.com/permalink.php?story_fbid=2407256322842204&id=100006735798590.
89. Menus Suck. https://www.gdcvault.com/play/1023668/Menus.
90. Test MRI display. https://web.stanford.edu/~kimth/brain/threejs/test_mri.html.
91. A beginners guide to understanding map contour lines | OS GetOutside. *OS GetOutside* https://localhosthttps://getoutside.ordnancesurvey.co.uk/guides/understanding-map-contour-lines-for-beginners/.
92. Website. http://hamishtodd1.github.io/VRC.
93. LEBARBA. http://web.archive.org/web/20160402041316/http://www.lebarba.com/blog/.
94. Volume - PyMOLWiki. https://pymolwiki.org/index.php/Volume.
95. Multi-contoured electron Density maps. *Anomalous Distraction* https://xtaldave.wordpress.com/2015/12/04/multi-contoured-electron-density-maps/ (2015).
96. Free Image on Pixabay - Mri, Magnetic, X Ray, Skull, Head. https://pixabay.com/en/mri-magnetic-x-ray-skull-head-782459/.
97. Johnson, J. E. Functional implications of protein-protein interactions in icosahedral viruses. *Proc. Natl. Acad. Sci. U. S. A.* **93**, 27–33 (1996).
98. Nanotechnology Art Gallery -- Accelrys. http://www.nanotech-now.com/Art_Gallery/accelrys.htm.
99. Wong, M. D., Dorr, A. E., Walls, J. R., Lerch, J. P. & Henkelman, R. M. A novel 3D mouse embryo atlas based on micro-CT. *Development* **139**, 3248–3256 (2012).

100. Tarini, M., Cignoni, P. & Montani, C. Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization. *IEEE Transactions on Visualization and Computer Graphics* vol. 12 1237–1244 (2006).
101. Xu, Z., Horwich, A. L. & Sigler, P. B. The crystal structure of the asymmetric GroEL-GroES-(ADP)7 chaperonin complex. *Nature* **388**, 741–750 (1997).
102. gamealchemist. Let's get those Javascript Arrays to work fast. *gamealchemist* https://gamealchemist.wordpress.com/2013/05/01/lets-get-those-javascript-arrays-to-work-fast/ (2013).
103. Campbell, M. G., Veesler, D., Cheng, A., Potter, C. S. & Carragher, B. 2.8 Å resolution reconstruction of the Thermoplasma acidophilum 20S proteasome using cryo-electron microscopy. *Elife* **4**, (2015).
104. Lee, J., Kim, J.-I. & Kang, L.-W. A Collaborative Molecular Modeling Environment Using a Virtual Tunneling Service. *Biomed Res. Int.* **2012**, (2012).
105. [No title]. https://www.ccp4.ac.uk/newsletters/newsletter49/articles/coot_xmlrpc.pdf.
106. *CppCon 2014: Mike Acton 'Data-Oriented Design and C++'*. (2014).
107. 2019 Model Metrics Challenge | EM Validation Challenges. http://challenges.emdataresource.org/?q=model-metrics-challenge-2019.
108. *VR Interface Design Pre-Visualisation Methods*. (2015).
109. Graft, K. The burrito that wraps up excellent VR design fundamentals. https://www.gamasutra.com/view/news/264648/The_burrito_that_wraps_up_excellent_VR_design_fundamentals.php.
110. Agarawala, A. & Balakrishnan, R. Keepin' it real. *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06* (2006) doi:10.1145/1124772.1124965.
111. Skeuomorphism is dead, long live skeuomorphism. *The Interaction Design Foundation*

https://www.interaction-design.org/literature/article/skeuomorphism-is-dead-long-live-skeuomorphism.

112. *Fantastic Contraption and why VR Menus Suck*. (2016).

113. Eifler, M. You're doing Mixed Reality wrong. *Medium* https://medium.com/@blinkpop/youre-doing-mixed-reality-wrong-d32aa54ae8af (2017).

114. Park, J., Long, X. & Lee, S. Virtual Reality Remote Pointing Experiment: Evaluation Using Fitts' Law. *Design Convergence Study* vol. 17 167–177 (2018).